A Blockchain-based online voting system for Belgium's elections.

Guillaume Wafflard

Thesis director :
Jean-Michel DRICOT

Member of the Jury :

Olivier MARKOWITCH

Christophe PETIT

MEMO-F524 Master thesis

Department of Computer Science,
Faculty of Sciences,

Université Libre de Bruxelles

January 7, 2024

Contents

Acknowledgements

I extend my gratitude to all those whose contributions and support made the completion of this thesis possible. First of all, I would like to acknowledge my thesis director, Jean-Michel Dricot, for his guidance, advice, and engaging anecdotes related to Belgian elections throughout the supervision of my thesis.

I would like to express my gratitude to Denis Verstraeten, who initially supervised me and agreed to follow my work and support me in exploring a topic at the intersection of democracy and blockchain technology — a field I am passionate about.

I also have a thought for Santiago Siri, the founder of Democracy Earth, whose conference on *Internet and blockchain-based technology, a revolution for Democracy* inspired me to the topic of this thesis.

I would like to acknowledge the vibrant community at the Université Libre de Bruxelles, a unique place that has taught me the profound impact each individual can have in the world. It is a place where people engage in diverse projects, uphold traditions, raise funds for charity, and strive to make the world a better place. I express my gratitude to the members of the Cercle Informatique, the activists of Extinction Rebellion ULB, and all the student associations that made my student year the most enriching of my life.

Finally, I want to thank my parents Ingrid and Didier, my brother, Quentin, and all my friends who supported me during my studies and my thesis, and my grandmothers, Lydia and Micheline, who never gave up trying to understand the title of my thesis.

Abstract

Voting with the Internet (i-voting) is often claimed as the future of elections. However, this method adds new risks in a domain where security is at the highest stake. Blockchain-based online voting systems could solve some of the issues of i-voting by bringing more transparency, integrity, and overall security thanks to its distributed architecture. In this paper, we first analyze the different voting systems (paper-based, electronic machines inside polling stations, and i-voting), and highlight the risk brought by each of them. Subsequently, we analyze six models of blockchain-based voting systems from the literature, distinct by their protocols and architectures, regarding the requirements of integrity, eligibility, anonymity, unreusability, availability, verifiability, coercion resistance, and scalability. Then, we propose our own model of blockchain-based online voting system, designed for hosting elections in Belgium. This system is based on the Hyperledger Fabric framework. The network connects peers from partner organizations selected for their cybersecurity expertise, and auditors or citizens can run read-only nodes for transparency. The eligible voters authenticate with the app itsme®. The votes are transmitted using the He and Su protocol [73] using blind signatures, and adapted to the distributed ledger by Kirilov *et al.* [86]. The model respects the requirements of integrity, authenticity, un-reusability, anonymity, freeness, auditability, verifiability, and availability, but it fell short in receipt-freeness. The eligibility remains on the election authority's responsibility. It is adapted to the specificities of the Belgian elections, and voters have the choice to vote either online or on-site.

Chapter 1

Introduction

Democracy, from the ancient Greek *dêmos* (the people), and *kratien* (to command), is a political system in which power is held by the people. Most of the Western countries are using this system. However, we can observe today in our society a loss of confidence of the population towards their leaders. This loss of confidence is generally felt in an increase in abstention, or in the rise of extreme political parties. Politicians and political analysts are actively searching for solutions to reconcile some branches of the population, especially the youth, with politics.

All the countries under democratic regimes apply the model of *representative* democracies. In a representative democracy, the citizens elect people to represent them, and these elected people then possess effective power, within the limit of their function and for a determined duration. In contrast, *direct* democracy, or «*participatory democracy*», is a system in which citizens directly participate in the writing of laws, or directly vote on laws themselves, without having an intermediary. An example of direct democracy is the ancient Greek city-states, where the citizens - the male inhabitants who were not slaves or metics (foreign residents) and who were over 20 years old with completed military service - directly voted on their laws without the need for representatives. In modern times, direct democracy can be applied through referendums, wherein the population is invited to vote on one particularly important decision. In Belgium, the referendum has no legal basis and may be considered unconstitutional. There is, however, a mechanism for "popular consultation" in this country, which consists of asking for the opinion of the population but without the result being binding[59]. This popular consultation varies from one federated entity to another and was used only once in 1950. On the other hand, Switzerland, for example, uses the term "votation" to describe the many occasions when citizens are called upon to express their opinions. If it may seem impractical to involve the entire population in voting on every law, one might wonder why elections and referendums are not held more often. Despite citizens voicing their opinions to friends, colleagues, and on social media daily, they are typically given the opportunity to formally vote only once every five years.

The most traditional voting system, used in many countries, is the paper ballot in a ballot box. In Belgium, voting is done using either paper or electronic voting machines, depending on the region. Popularized since the 1990s, this last method, consisting of voting with electronic machines, inside the voting booth in the polling stations, is called *e*-voting. The system of e-voting varies from one country to another, from simple electronic support for counting to direct recording machines (DRE) managing both the casting and tally of the votes.

A more recent voting system, already in use in Estonia since 2005 and in local elections in several other countries, is Internet voting, or I-voting. In our interconnected world, with over 90% of the European population and 5 billion people worldwide connected to the Internet [52], online voting might appear as the voting system of the 21st century. In this system, voters can vote remotely from their computer's browser or their mobile phone, from wherever they are located, on the only condition they have the right to vote and are connected to the Internet. However, electronic and online voting have been active topics of research in the last decades, and scientists have not

managed to find a perfect solution. Elections are the fundamental basis of democracies, and one of the pillars of elections is vote secrecy. No other topic on the internet requires such a high level of privacy than online voting. Not even financial transactions (where the bank at the origin of the transfer knows the identity of the sender and the recipient), nor confidential web browsing, where internet providers and hosting servers collect metadata that could reveal some information of the user's identity. Online voting also requires an extreme level of security against a corrupted entity or a foreign threat. Some characteristics defining an ideal internet voting system even conflict with each other, like the opportunity for a voter to verify his or her [1] vote clashing with the essential requirement to prevent the voter from proving how he or she voted to anybody else.

After defining the list of requirements specific to each type of voting system (Section 4.1), and exploring these different systems used around the world, their specificities, and the advantages and drawbacks of each, we will evaluate what the literature says about online voting, its risks, and its limitations (Section 4.2). Then, this paper focuses on a sub-category of Internet voting: Internet voting based on blockchain technology (Section 4.3). The main advantages sought by this solution are integrity, transparency, and robustness. With a publicly readable blockchain, the results of the election could be verified by everyone, as all the votes are recorded in this public database, but encrypted to guarantee the voters's anonymity. The tamper-proof property of a blockchain prevents any alteration of the votes, and some level of decentralization will enhance the service robustness compared to single-server alternatives. After analyzing how each election requirement and I-voting specifics requirement adapts to this technology, we will explore the models of blockchain-based e-voting systems proposed in the literature, and analyze in-depth 6 distinct models with original protocols and architectures in Section 5. Finally, after a comparison of these models based on objective criteria (Section 5.3), this paper will propose in Section 6 an adaptation of the best model to fit Belgium's tools and specificities. This solution will be compared against the requirements established earlier (Section 7.1). We terminate in Section 7.3 with a discussion on the consequences that this system would have if ever implemented for real elections, regarding the cost and the environmental impacts of organizing online elections, the democratic possibilities it offers, and legal and societal considerations.

[1] In rest of this thesis, the use of feminine pronouns (she/her) or the less frequently the singular gender-neutral pronouns (they/them) to describe an arbitrary user or voter is a deliberate choice to promote gender inclusivity and counter the default use of masculine pronouns in academic discourse.

Chapter 2

State of the Art

Different voting systems

This paper will analyze and compare different voting voting systems used around the world. These systems are traditional/paper-based, e-voting with DRE and VVPAT, and i-voting. Each of these systems will be given some context and described shortly in Section 4, but they are all well-documented in other papers. Dimitris A. Gritzalis describes in detail the voting process in the traditional paper-based system in [65]. Halderman explains the configuration and the voting process using direct-recording electronic (DRE) machines in [54]. Villafiorita, Weldemariam, and Tiella provide in *Development, Formal Verification, and Evaluation of an E-Voting System With VVPAT* [137] a detailed description of an e-voting process with a voter-verified paper audit trail (VVPAT, see Section 4.2.3). Specifically, they analyze and do a formal verification of an e-voting system, called ProVotE, which is an end-to-end verifiable e-voting system with a VVPAT, that is sponsored by the Autonomous Province of Trento in Italy. The papers *A review of E-voting: the past, present and future* [61] by Paul Gibson *et al.* and [138] by Maria-Victoria *et al.* list many countries using i-voting or developing pilots to do so.

Election and online voting

Estonia was the first country to use internet voting for its national election [112, 47]. Online voting is not just a recent trend. It has been studied in the literature for decades. In particular, the security aspects of online elections are a recurring domain of research.

Multiple authors have formalized the security requirements for an online election. Md. Abdul Based in *Security Aspects of Internet based Voting* [16] cites as security requirements the authentication and identification of the voter, the ballot encryption and signing, the encrypted ballot transmission over the Internet, the privacy of the voter, the anonymous ballot decryption, and the counting of ballots, all in a secure way. In *Security Requirements for Internet Voting Systems* [17], Md. Abdul Based and Stig Fr. Mjølsnes consider the basic information security requirements for Internet voting systems as the eligibility of the voter, the confidentiality and integrity of the ballot, the privacy of both the voter and the ballot, the robustness and availability of all voting system and its components, and the fairness, the soundness, the completeness, and the unreuseability of the counting process. They also consider as "expanded requirements" the validity, the verifiability, the receipt-freeness, and the coercion-resistance. In *Development of A Formal Security Model for Electronic Voting Systems* [23], Katharina Bräunlich and Rüdiger Grimm cite fairness, eligibility, secrecy, receipt-freeness, and universal verifiability as the necessary requirements to hold an online election, along with the possibility for a voter to modify its choice to prevent coercion.

Numerous other papers, some of which are referenced later in this document, mention similar requirements. While the specific requirements and the terminology used to describe them may differ slightly from one paper to another, the recurring main concepts provide a clear understanding of the essential needs. The most common properties considered as necessary to host online voting cited in the vast majority of

the paper are integrity, anonymity or confidentiality, authenticity or eligibility, availability, verifiability, and security, even though the specific terms used to describe these requirements might differ. A comprehensive enumeration and explanation of all these requirements can be found in Section 4.1.

J. Benaloh and D. Tuinstra [20] discussed the concept of *receipt-freeness* of an election. They explain why an elector should not be able to get a receipt of their vote, in order to avoid vote buying and coercion.

The concept of individual and universal verifiability for e-voting is formally defined by Steve Kremer, Mark Ryan, and Ben Smyth in *Election Verifiability in Electronic Voting Protocols* [87] and Rui Joaquim, Paulo Ferreira, and Carlos Ribeiro extend this concept for Internet Voting in *EVIV: An end-to-end verifiable Internet voting system* [79].

e-voting models

In the paper *An Internet Voting System Supporting User Privacy* published in 2006 by Aggelos Kiayias *et al.*, authors identified three cryptographic design approaches for e-voting systems at the time of their writing: *mixnet-based* (see Section 3.1.6), *blind-signature-based,* (see Section 3.1.3) like A. Fujioka, T. Okamoto, and K. Ohta in 1993 [57] or O. He and Z. Su in 1998 [73], and *homomorphic encryption-based.* In the latter approach, homomorphic encryption's capability to perform computations on encrypted data is utilized. This allows the election results to be calculated using encrypted ballots. The final tally is then decrypted at the end, eliminating the need to decrypt each individual ballot. This method was proposed by J. Benaloh in 1987 in his PhD thesis *Verifiable Secret-Ballot Elections* [21] that is still considered today as a reference in the domain of verifiable voting systems. Jens Groth used another cryptographic design, using Non-interactive Zero-Knowledge Arguments for Voting [66] (see Section 3.1.4).
Helios [6] is an online voting system that was based on mixnets in its first version and, in Helios 2.0, mixnets were replaced by homomorphic encryption. It is designed for scenarios where privacy and election integrity are important, but the risk of coercion is low. It is ideal for environments such as student elections, and online community decisions, where ensuring secure and private voting is essential, yet the likelihood of undue influence or pressure on voters is relatively low [113]. It has been used for the election of board members for the International Association for Cryptographic Research (IACR) [70].

Risks of e-voting and i-voting

Halderman is a specialist in risks with e-voting (voting on electronic machines inside the voting stations) and i-voting (voting by internet). In *Practical Attacks on Real-World E-Voting* [72], he lists the main risks related to e-voting. These risks are discussed in section 4.2.2. In *Security analysis of the Diebold AccuVote-TS voting machine* [54], Halderman and other authors explain how they could run malware on Diebold accuvote-ts voting machines, some Direct Electronic Recording machines widely used in the US. In 2010, a team from the University of Michigan, which he led, compromised a Washington D.C. Internet Voting system during its testing phase, just before its official launch. The specifics of this attack and the vulnerabilities uncovered are elaborated in [140]. C. Enguehard, as well as Sunoo Park, Michael Specter, Neha Narula, and Ronald L. Rivest (co-inventor of RSA), have expressed concerns in their respective studies [50, 106] that vote manipulation, traditionally limited in scale and more observable in conventional voting systems, could potentially grow to a much larger scale and become more challenging to detect in the context of online voting.

Blockchain applications

The concept of "blockchain" gained prominence with its implementation in Bitcoin, a cryptocurrency system introduced in 2008 by an anonymous entity or group known as

Satoshi Nakamoto, in the white paper "Bitcoin: A Peer-to-Peer Electronic Cash System" [100]. While the term "blockchain" was popularized by Bitcoin, some underlying cryptographic techniques and data structures predate it. This is the case for instance for the Hashcash [15] system, a Denial of Service counter-measure invented by Adam Back in 1997 that introduced Proof-of-Work; the concept of chains of block developed by Stuart Haber and W. Scott Stornetta for timestamping documents [71]; and Merkle trees [96]. Early ideas of decentralized digital currencies like Wei Dai's b-money [37] and Nick Szabo's Bit Gold [126] have served as inspiration for Bitcoin, but neither system was fully implemented. In the years following the launch of the Bitcoin network, numerous cryptocurrencies based on blockchain technology emerged (e.g. Ripple [28], Ethereum [24], Monero [134], Dash [43], Peercoin [84], ...), challenging traditional banking by eliminating the need for a central authority. In more recent years, the application of blockchain has expanded significantly, with developments in both industry and academia extending well beyond just cryptocurrencies [64]. Some applications are related to the financial sector [97], such as banking services [68], Decentralized Finance (DeFi), and digital assets like Non-Fungible Tokens (NFTs). Additionally, blockchain technology is increasingly being applied in supply chain management and logistics [108], the Internet of Things (IoT) [122], decentralized cloud computing and data storage [142], and cybersecurity [141]. In the energy sector, it can be used for innovations such as peer-to-peer energy trading [11]. Furthermore, blockchain is proving valuable in the secure storage and sharing of sensitive documents, a feature particularly useful in sectors like education (see [31] and Section 3.5), healthcare [114], and many more.

Benefits, Risks and limitations of blockchain-based e-voting system

Another type of emerging application of blockchain is secured remote voting system. The paper [98] by Moura *et al.* highlights that blockchain technology offers several advantages for voting systems, including transparency, immutability, high availability, reliability, auditability, and increased voter confidence, thanks to its trustless, distributed, and decentralized architecture. However, it also notes that blockchain technology has its drawbacks. To facilitate communication with other nodes in a peer-to-peer network, voting systems must be connected to this network. This connectivity exposes them to potential cybersecurity issues and threats.

Patrick McCorry *et al.* highlight in *On Secure E-Voting over Blockchain* [94] the security advantages of blockchain over a centralized server. They note the risk of data manipulation with a single server and suggest using mirrored websites for data replication to prevent unnoticed alterations. However, they propose blockchain as a more effective solution, as it synchronizes replicated ledgers across a network of peers, ensuring consensus and enhancing data integrity.

Hyunyeon Kim et al. [83] and B. Shahzad and J. Crowcrof [120] also argue in the same direction, saying that blockchain improves the online voting system by adding transparency, integrity, and security.

On the other hand, the paper *Going from Bad to Worse: From Internet Voting to Blockchain Voting* by Sunoo Park *et al.* expresses skepticism about the effectiveness of Internet- and blockchain-based voting systems, arguing that they could significantly increase the risk of large-scale, undetectable election failures. It emphasizes that auditability alone is insufficient; it must be paired with actual auditing to ensure election integrity. The paper compares this to the practice of collecting receipts for credit card expenses but never actually reviewing them. Furthermore, it acknowledges that even traditional methods like paper ballots aren't completely fail-proof, especially if election officials manipulate the process in secrecy. To mitigate such risks, the paper highlights the importance of transparency measures, such as allowing independent observers to monitor the election process. In their paper, they admit that no system is perfect, as even paper-based voting has its own vulnerabilities. However, they consider that Internet voting is too risky, and that blockchain-based voting will not solve the issues.

S. Gandhi *et al.*[58] analyzed the security requirements of blockchain-based E-Voting Systems. It established a list of Functional Requirements, non-functional requirements,

and countermeasures to typical attacks.

Literature reviews on blockchain-based e-voting systems

In *Blockchain for Electronic Voting System—Review and Open Research Challenges* [78] by Uzma Jafar *et al.*, the authors analyze 6 commercial solutions for blockchain-based voting (Follow My Vote, Voatz, Polyas [104], Luxoft, Polys, and Agora [8]) and conclude that none of them are scaling enough for large-scale elections. They also analyzed 6 models proposed in the literature by academics. They consider that most of them do not scale enough, and for those where scalability is not an issue, other requirements are not totally respected, such as an insufficient guarantee of anonymity, or the impossibility of auditing the system.

In a study named *E-Voting Meets Blockchain: A Survey* [138] conducted by Maria-Victoria Vladucu, Ziqian Dong, Jorge Medina, and Roberto Rojas-Cessa, a comprehensive comparison of various e-voting systems utilizing blockchain technology is presented. The research first compares eight distinct consensus algorithms, including Proof of Work, Proof of Stake, Delegated Proof of Stake, and Practical Byzantine Fault Tolerance (PBFT), among others. Then, the paper evaluates six blockchain frameworks based on several criteria: the consensus algorithm employed, block generation time, access level (public or private), transaction rate, and scalability. The findings suggest that private blockchain solutions like Exonum, Hyperledger-Fabric, and Quorum exhibit greater scalability compared to public blockchains reliant on Proof of Work (PoW) consensus protocols. Furthermore, the paper conducts a literature review of various models of e-voting based on blockchain, categorizing them based on their blockchain framework, their focus on specific features (such as integrity, anonymity, or verifiability), and their registration methods, labeling them as 'internal' if the registration process is integrated within the blockchain system, or 'external' if registration is managed by a trusted third party or through verifiable databases provided by governmental or authoritative entities. The classification also notes whether each model is an implemented solution or remains theoretical. The paper categorizes each model by its main feature (requirement) but does not provide a comparative table of whether and how each requirement is respected. It is therefore not straightforward to conclude whether a model could fit our use case or not. Additionally, the paper comments on 13 commercial e-voting solutions, offering a brief description of each and noting their real-world applications. However, it does not evaluate these commercial solutions against specific requirements.

The paper *A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting* [129] by Ruhi Tas and Ömer Özgür Tanrıöver categorizes the prevalent challenges in blockchain-based e-voting into five areas: general, integrity, coin-based, privacy, and consensus, based on a review of 63 research papers. It considers that blockchain technology can address some existing issues in election systems. However, the study also identifies key concerns with privacy and transaction speed in blockchain frameworks. It emphasizes the need for enhancing security for remote voting and scalability to ensure the sustainability of blockchain e-voting. The conclusion suggests that these frameworks must be improved before they can be effectively implemented in voting systems.

Belgian elections

C. Vegas[136] explains the history of the electronic vote in Belgium from 1991 to 2012. The paper explains the first generation of polling machines, and the criticisms they faced, resulting in the second generation of polling machines that has been in use since 2014 until now. The paper also describes this "new" system and compares it with the international standards for elections. R. Dandoy [38] has studied the impact on the turnout of e-voting against paper voting in Belgium's local election, despite mandatory voting.

The report of the experts in charge of the control of the IT systems of the 2019 election [45] provides a detailed explanation of the process of the election (detailed in Section 3.4). The authors describe the hardware and software used during the elections and developed by the private companies Martine and SmartMatic. They dress a list of issues observed during their control and make recommendations to improve the system in the future.

A study on the possibility of introducing Internet voting in Belgium [132] was conducted by a consortium of Belgian universities for the Federal Public Service of the Interior. The study is divided into two phases. The first part, published in 2020, analyzed past and present experiments of Internet voting in 5 distinct countries: Australia, Estonia, France, Norway, and Switzerland. For each country, they analyzed four key dimensions: Computing and Security, the acceptance by citizens and authorities, the logistical organization, and the legal and regulatory aspects. They conclude by establishing a list of recommendations for the design of a new system of Internet voting. The second part of the study, published in 2021, evaluates the possibility of introducing mail-in voting including online elements for citizens living abroad. This evaluation is based on the four same dimensions used for the analysis in the first part of the study. It proposes a model of verifiable mail-in ballots that would be more efficient than the current mail-in voting system.

Chapter 3

Background

3.1 Cryptography

3.1.1 Hash function

A hash function is a deterministic function that maps an arbitrary size input to an output of fixed size (like 256-bits). A hash function is a cryptographic hash function if it is a one-way function and collision-resistant.[128]
One-way function means that it is easy to compute (*on input x, there is an efficient algorithm that outputs f (x)*) and hard to invert (*given f (x) as an input, any efficient algorithm that tries to invert f (to find the preimage of f (x)) succeeds only with negligible probability.*).
Collision-resistant means that *for a hash function H with a security parameter n, the probability of finding a pair of inputs value x,y where $H(x) = H(y)$ is a negligible function on the security parameter n.)*

Random Oracle

A Random Oracle is an idealized version of a cryptographic hash function, where the output seems obtained completely randomly. When a protocol's security depends on the hash function's primitive, the protocol can be declared *secured in the random oracle model*, meaning that it is considered secure only with the assumption that the hash function is perfectly random.

3.1.2 RSA

RSA [115], named after the initials of its inventors, Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, is an asymmetric public key cryptosystem. RSA allowed two innovative cryptographic concepts: an asymmetric encryption that permits private sharing of messages without the need of securely sharing a symmetric key, and a digital signature proving the authenticity and integrity of a message. Imagine having two agents, Alice and Bob, who want to communicate with each other on an unsecured network. The general idea is to have a public key, that can be - as its name indicates - publicly shared, and a private key that must be kept secret. Alice can encrypt a message m to Bob using Bob's public key $E_B(m)$, and Bob can decrypt this message using its private key

$$D_B(E_B(m)) = m$$

Alice can also sign a message with her private key, to prove that she is the real sender of the message and that the message hasn't been altered: $S = D_A(m)$. Bob can then verify the signature with Alice's public key.

$$E_A(S) = E_A(D_A(m)) = m$$

Here is how it works:

Key Generation

1. Select two distinct large prime numbers p and q.
2. Compute $n = pq$. This product n is used as the modulus for both the public and private keys.
3. Calculate Euler's totient function, $\phi(n) = (p-1)(q-1)$.
4. Choose a public exponent e such that $2 < e < \phi(n)$, and e is coprime to $\phi(n)$. Often, $e = 65537$.
5. Determine the private exponent d to satisfy $d \cdot e \equiv (1 \bmod \phi(n))$. d is coprime to $\phi(n)$. d is the modular multiplicative inverse of e mod $\phi(n)$.

Encryption

- The public key is made of the pair (n, e)
- To encrypt a message m (where $m < n$), the ciphertext c is computed as $c \equiv m^e \bmod n$

Decryption

- The private key is composed of the pair (n, d)
- To decrypt the ciphertext c, the original message m is recovered by computing $m \equiv c^d \bmod n$

Therefore, this equality shows that decrypting an encrypted message m leads to m: $D(E(m)) \equiv (E(m))^d \equiv (m^e)^d \pmod n = m^{e \cdot d} \pmod n = m$

3.1.3 Blind signature

Blind signatures are a type of digital signature proposed by David Chaum [29] in 1983 with the specificity that the signer doesn't know what it signs. Let's take the same actor as before. Bob (a user) has a message m that he wants to get signed by Alice (a signer), but without revealing the actual contents of m to her. The process involves the following steps:

1. Bob applies a blinding function B to the message m : $m' = B(m)$
2. Alice signs the blinded message m' with her private key, producing the blind signature : $S(m')$
3. Bob receives the blind signature $S(m')$ and applies an unblinding function U to it, which reverses the effect of the blinding factor. The result is a valid signature $S(m)$ of the original message m that can be publicly verified with Alice's public key.

The blind signature satisfies the properties of *Unlinkability* (t is computationally infeasible to associate a blinded message-signature pair with its corresponding unblinded pair), *Non-forgeability* (Only the signer (Alice) can produce valid signatures, ensuring the integrity and authenticity of the signed message), and *Blindness* (The signer (Alice) cannot determine the original message m from the blinded message m').

3.1.4 Zero-Knowledge Proofs (ZKPs)

Zero-Knowledge Proofs were first conceived in 1985 by S. Goldwasser, S. Micali, and C. Rackoff in [63]: *"Zero-knowledge" proofs allow one party (the prover) to prove to another (the verifier) that a statement is true, without revealing any information beyond the validity of the statement itself.* For example, a user (prover) could mathematically prove to a server (verifier) that he/she knows the password, without revealing it.

3.1.5 Non-interactive Zero-Knowledge Proof

Non-Interactive Zero-Knowledge Proofs (NIZKPs) [22] are ZKPs that do not require interaction between the prover and the verifier. The prover computes a static one-directional proof that can be verified by any verifier in possession of the proof, without

the need of asking any challenge. NIZKPs are highly beneficial in blockchain contexts. They enable a prover to record a proof on the blockchain, allowing any verifier reviewing the blockchain data to efficiently validate the proof later [128].

3.1.6 Mixed Network (mixnet)

Mixed Networks (Mixnets) are routing protocols invented by D. Chaum in 1981 in [30] designed for anonymizing the source and destinations of online communications. They employ a series of intermediary nodes, known as mixes, which collectively hide the communication paths within a network. These nodes receive batches of encrypted messages, decrypting and re-encrypting them before forwarding them in a reshuffled order. This process prevents any linkage between incoming and outgoing messages, making the communications untraceable.

3.1.7 Identity Mixer (idemix)

Identity Mixer (Idemix) is a cryptographic protocol designed to authenticate while preserving privacy. It allows users to confirm their membership in a group or their credentials without revealing their full identity. This is achieved using zero-knowledge proofs (ZKPs), which enable users to demonstrate the possession of specific credentials or attributes without disclosing any extra information. As anonymity is protected, multiple transactions by the same identity are not traceable back to that identity. Idemix involves three main actors: the user, the issuer, and the verifier. The issuer provides the user with a set of credentials in the form of a digital certificate. When the user needs to authenticate or demonstrate certain attributes, they produce a zero-knowledge proof derived from these credentials. This proof reveals only the necessary attributes to the verifier, preventing any additional unnecessary information from being exposed.

3.2 Blockchain technology

In this section, we will define some notions related to blockchain technology. We recommend the reader to be familiar with it before getting into the next parts.
The term *blockchain* has been popularized by the inventor of the first cryptocurrency under the pseudonym of Satoshi Nakamoto in 2008 in the Bitcoin white paper «*Bitcoin: A Peer-to-Peer Electronic Cash System*»[100], even though the data structure was invented earlier (cf. State-of-the-Art). This term has become a "buzzword" over the years and is assigned several meanings. We can define blockchain as a *distributed ledger technology* (DLT) implemented on a P2P network [138]. We will describe it first from a data structure perspective, and then from a network perspective.

3.2.1 Blockchain data structure

A blockchain is an append-only data structure, where data is stored in a distributed ledger that cannot be tampered with or deleted. All the transactions are recorded inside blocks that are chained one after the other (from there comes the term "block-chain") in such a way that each block contains an entry that is a hash of the previous block [75]. A widely used hash function is SHA-256, which belongs to the family of SHA-2 hash functions, designed by NSA and standardized by NIST[56]. Therefore, if any entry of a block is modified, the hash of the next block wouldn't correspond to this "modified block", and the chain would become invalid. Figure 3.1 shows a visual representation of such a chain of blocks.

3.2.2 Blockchain network

What makes the blockchain a secured ledger is that it is distributed. This means that the blockchain is copied in the exact same state in all the nodes of the network. When

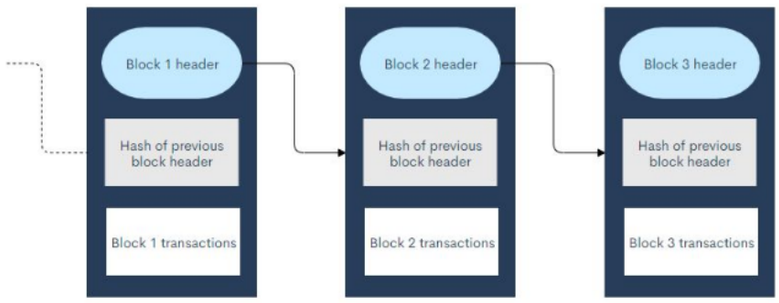


Figure 3.1: Representation of the blockchain as a chain of blocks, where each block contains a hash of the previous block. From [40].

a node creates a new block, it broadcasts it to all the nodes of the network, and these nodes add the block - after verification of its validity - at the end of the blockchain, preserving the similarity property of the distributed blockchain. The set of rules that determines whether a node should accept or not a new block is called the *consensus algorithm*. If one node gets hacked or disconnected, the other nodes of the network remain and the blockchain is still active. Bitcoin (used as an example because it was the first cryptocurrency, and its design inspired all other blockchain projects after it) was designed as a Peer-to-Peer (P2P) electronic cash system. The idea behind that is that a user can send money to another user without involving any third party such as a financial institution. The only intermediary is the blockchain network, and the system is said *trustless* because it doesn't need any trust in anybody in the process. All the trust remains *in the algorithm*.

Blockchains can be distinguished based on network accessibility and transaction validation permissions:

- **Public** Blockchain: These blockchains are open to the public. Anyone can read the blockchain, submit transactions, and participate in the consensus process.
- **Private** Blockchain: Access to these blockchains is restricted to a select group of nodes. They are typically operated by private organizations and offer greater control over participants.

Regarding transaction validation, blockchains can be divided into two categories :

- **Permissionless** Blockchain: This is a fully decentralized model where anyone can participate in the transaction validation process.
- **Permissioned** Blockchain: In this model, transaction validation is controlled by a central authority or a predefined mechanism that authorizes only certain nodes to validate transactions. This setup can be found in both private and public blockchains. Public permissioned blockchains allow anyone to read and submit transactions, but only authorized nodes can validate them. Permissioned blockchains generally offer better scalability due to the ease of synchronization among fewer validators, resulting in shorter times between blocks and higher transaction throughput.

3.2.3 Consensus Protocol

A Consensus Protocol is a set of rules followed by all the nodes of the network to agree on the state of a distributed ledger, i.e. to accept new blocks. Consensus Protocols are particularly required for decentralized permissionless blockchains where they ensure that all honest nodes maintain a consistent view of the blockchain, despite the ability for dishonest nodes to freely join and participate in the network.

Proof-of-Work (PoW)

Inspired from Hashcash [15], Proof-of-Work was used in the first generation of cryptocurrencies allowing for a fully decentralized system without central authorities. It is

based on the idea of *one-CPU-one-vote* [100]. In this protocol, miners (nodes dedicating computational resources to generating new blocks) group multiple transactions into a single block. Their primary task is to find a specific number, known as a nonce, that, when combined with the block's data and passed through a hash function, produces a hash value that is inferior to a predefined threshold. This threshold is dynamically adjusted based on the network's average computational power over the past two weeks, ensuring a consistent rate of block creation. For example, if the network requires the hash value to start with 20 zeros (when represented in hexadecimal format), the probability of a miner finding a valid hash in a single attempt is $1/16^{20}$. This incredibly low probability necessitates an important amount of computational effort and numerous trials, making the mining process resource-intensive. Once a miner successfully found a nonce that meets these criteria, it has effectively 'mined' a new block. This block is then broadcast to the rest of the network. Other nodes in the network will verify the validity of the block by checking if the hash meets the difficulty criteria. If it does, the block is accepted and added to the blockchain. The other nodes must always accept the longest chain (i.e. containing the most blocks).

The integrity of the network is maintained as long as the majority of its computational power—over 51%- is controlled by honest nodes. In a scenario where dishonest participants aim to reverse a transaction, such as to execute double-spending, they would need to accumulate enough computational power to not only invalidate a specific block but also construct a longer and valid blockchain alternative. In PoW, miners are incentivized to contribute their computational resources and respect the consensus protocol by the coinbase reward. This reward consists of newly minted tokens, issued to financially compensate nodes for their participation in the network. Miners also earn revenue from transaction fees (see below), further incentivizing their commitment to maintaining the blockchain.

Unfortunately, this protocol based on trial and error is energy-inefficient. Nowadays, as miners are using a huge quantity of computational power to secure the network, it has consequences to waste a tremendous amount of electricity. The Bitcoin network has by itself an important impact on the environment. Despite the exact impact being impossible to calculate, C. Stoll *et al.* [124] estimated the annual electricity consumption of Bitcoin in 2019 up to 45 TWh and the annual footprint in a range from 22.0 to 22.9 MtCO2. Since then, the Hash rate (the amount of hashes computed per second) has been multiplied by a factor of 10, reaching 500 million TH/s in January 2024.

Proof-of-stake (PoS)

In Proof-of-Stake protocol, nodes owners must put at stake a quantity of money (i.e. a number of tokens of the native cryptocurrency) to become a Validator. Then, for each block, one Validator is randomly selected by the protocol to create the next block. A Validator in a PoS protocol is incentivized to act honestly (i.e. respecting the rules of the protocol) by receiving a reward for blocks it created honestly, or losing a part of its stake if it creates invalid blocks. Ethereum uses PoS as consensus protocol since September 2022.

Many other types and variants of consensus protocol have been proposed and used in various projects. **Raft**, a crash fault-tolerant (CFT) ordering protocol, and an implementation of a **Practical Byzantine Fault Tolerant** (PBFT) are presented later in Section 3.3. The purpose of this section is not to provide an extensive list but to explain the main ideas behind them. Note that private and permissionless blockchains also have a consensus protocol, but those are usually simpler, as they are traditionally managed by organizations trusting every node.

3.2.4 Ethereum Virtual Machine (EVM)

Whereas Bitcoin's only use case is to send money from one address/user to another, Ethereum, on its side, lets a user store any type of data and run computation *on-chain*.

These computations are done on the *Ethereum Virtual Machine (EVM)*, a stack-based virtual machine that has its own Turing-complete programming language, a bytecode language referred to as *EVM code*[128]. Ethereum network was launched in 2015 [24] and its native cryptocurrency, *ether*, is the second biggest cryptocurrency by market capitalization, after Bitcoin.

3.2.5 Smart Contract

A smart contract is a code snippet that is described in EVM language and stored in the Ethereum blockchain [128]. In simpler words, a Smart Contract is a *program*, stored and executed on the blockchain, that contains internal variables and also functions that can be called by the address of users and/or other Smart Contract and/or the Smart Contract itself, using a *transaction*. The name Smart Contract might be confusing. It comes from the fact that the program will execute predefined rules if the conditions are met, as would do the two parties of a signed contract or a third party such as a notary. For example, rules can be *"Send `ether` to this person at that time if this condition is met"* or *"If the person who called the function of this smart contract is the owner of a certain `NFT` in this other Smart Contract, then execute this operation"*. The system is *trustless* because there is no need to trust any of the parties of an operation, the outcome of the Smart Contract will always be as described by the algorithm.

3.2.6 Gas and fees

To motivate a validator on a permissionless blockchain network to process a transaction, users pay an incentive known as *fees*. These fees can be likened to fueling a car with gas; the amount of *gas* required depends on the computational work involved in the transaction. When there's a high demand for processing transactions – exceeding the network's capacity to handle them simultaneously – users may increase the gas price they are willing to pay to ensure their transactions are prioritized. The total fees a user pays depend both on the computational effort required (gas used) and the network demand at the time of the transaction. These fees are paid in the native cryptocurrency of the blockchain; for Ethereum, this is in *ether*. The formula for calculating transaction fees is as follows:

$$Transaction fee = Gas limit * (Base fee + Priority fee)$$

In this formula:

- `Gas Limit` refers to the maximum amount of computational work (gas) that the transaction can consume.
- `Base Fee` is the cost per unit of gas at the time of the transaction, determined dynamically by the network based on demand.
- `Priority Fee` is an optional extra amount a user can pay to expedite their transaction, particularly useful during times of high network congestion.

These fees are often in *gwei*, the smallest unit of Ether (ETH) on the Ethereum network. One *gwei* equals $10^{-9} eth$

3.2.7 Decentralized application (dApp)

A decentralized application (dApp) is a type of application that operates on a decentralized network, typically a blockchain, rather than relying on centralized servers. It typically consists of a front-end user interface, such as a mobile app or a website, and a back-end in the form of smart contracts that run on a blockchain. The use of blockchain as the back-end framework eliminates the need for centralized server maintenance, transferring the computational and storage tasks to the nodes within the blockchain network. However, this decentralized architecture means that any computation or memory modification on the blockchain incurs transaction fees at the charge of the user. For instance, dApps enable developers, especially those building applications involving financial operations or similar functions, to focus on their core development

without the need for extensive security concerns or reliance on traditional banking services.

3.2.8 Decentralized Autonomous Organizations (DAOs)

Decentralized Autonomous Organizations (DAOs) are entities operating based on smart contracts, which automate their rules and operations, eliminating the need for a centralized governing body. Participation and governance in DAOs are often linked to holding native tokens, granting members voting rights and influence proportional to their token holdings. Governance in DAOs is democratic, with proposals ranging from protocol updates to fund allocations being submitted by members. The decision-making process is transparent, with each member's vote usually weighted by their token stake. Votes are cast during predefined periods, and proposals require meeting specific quorum and majority conditions to pass. This system ensures transparency and collective decision-making, aligning with the ideals of decentralization. However, a challenge arises in some DAOs where token distribution is disproportionate. In such cases, a few entities holding a major portion of the token supply can handle significant, potentially anti-democratic power within the organization.

3.2.9 The blockchain Trilemma

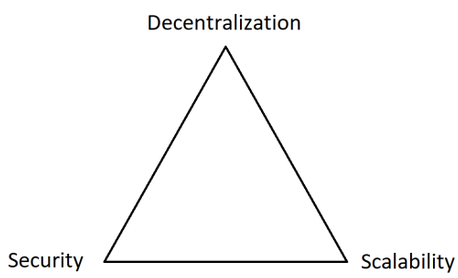


Figure 3.2: The Blockchain Trilemma. No perfect blockchain can optimize at the same time decentralization, security, and scalability. Usually, optimizing two of these characteristics is done at the expense of the third one.

One of the reasons why there exists no "perfect blockchain" that would have overthrown all the others, is because no blockchain was (yet) able to maximize at the same time *decentralization*, *security* and *scalability*. This problematic is called the *Blockchain Trilemma*.
Decentralization : Decentralization describes the absence of central power on the blockchain. The network is controlled by all the users and no person or institution can decide or modify anything alone.
Security : In this case, security is the ability for a network to resist node failures or hacks, and the difficulty for an opponent to take control of the blockchain.
Scalability : The capacity for the network to grow and continue to offer the same speed and amount of transactions per second.
For example, the more a network is decentralized and secure, the more nodes need to synchronize without a conductor. E.g. the Bitcoin network is widely decentralized and extremely secured, but the transaction speed is slow (there is only one new block every 10 minutes on average, with a throughput of 7 transactions per second). To increase the scalability, there is usually a sacrifice of a bit of security or decentralization. For example, the Binance Smart Chain is a blockchain with high speed and high transactions per second, but that is highly centralized, running on a few dozen nodes, and most of them are held by the private company Binance itself.

This dilemma has a parallel in the CAP theorem, or Brewer's Conjecture [62], for distributed databases, stating a system can only provide two of three properties: *Consistency, Availability,* and *Partition Tolerance*. Consistency ensures uniform data

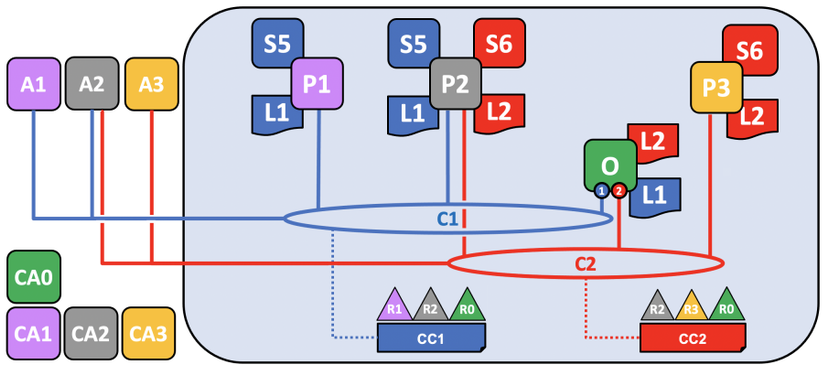


Figure 3.3: A diagram representing an example network in Hyperledger Fabric. Source: the Hyperledger Fabric's official doc [53]

across nodes, Availability ensures nodes are always functional, and Partition Tolerance ensures operation despite network failures. Both concepts illustrate trade-offs in distributed database systems, but it's important to note they are not identical and apply to different aspects of distributed systems.

3.3 Hyperledger Fabric

Launched in December 2015 by the Linux Foundation, Hyperledger [139] is an open-source project dedicated to the development of blockchain and distributed ledger technologies. Hyperledger is not a single blockchain entity; rather, it's an ecosystem of compatible frameworks, instruments, and libraries for a business-oriented use of blockchain technology. Among the multiple projects composing the Hyperledger ecosystem, *Hyperledger Fabric* is a permissioned blockchain network that offers modularity for a wide range of industry use cases and is designed to support pluggable implementations of different components.

A Hyperledger Fabric network is characterized not simply as a collection of nodes maintaining a single blockchain. Rather, it presents a complex network structure, potentially comprising numerous channels, where each channel maintains its distinct ledger and executes smart contracts. These are managed by peers who are part of various organizations, with their specific rights and privileges governed by the policies of each channel. To provide a global overview of the Fabric's architecture and its main difference with traditional blockchain design, we will now explain some concepts.

Networks stakeholders

A basic Fabric network is depicted in Figure 3.3. This network is composed of two distinct channels (C1 and C2) and four organizations (R0, R1, R2, and R3), each of them running one node (respectively O, P1, P2, and P3). Organizations are the primary entities that own and operate the network components. Each organization has an associated Certificate Authority (CA), such as CA0, CA1, CA2, and CA3 in the diagram. These CAs are responsible for issuing digital certificates like x.509 that authenticate the identity of nodes and clients within the organization.

Peers, such as P1, P2, and P3, are network nodes that serve to maintain a copy of the ledger (L1, L2) and execute and maintain the state of smart contracts (chaincodes) like S5 and S6. These peers are owned by organizations and endorse transactions or commit them to the ledger.

Channels, depicted as C1 and C2, are communication mechanisms working as private sub-networks within the global Fabric network, allowing a subset of authorized organizations to communicate and conduct transactions privately. Each channel has its distinct ledger and chaincodes. For instance, the ledger L1 may be associated with channel C1 and maintained across multiple peers, while ledger L2 is associated with channel C2. Organizations have collectively agreed-upon policies. These policies dictate the network's evolution and the roles that each organization plays within it. An

organization can leave or join a channel according to the channel's policies. Channels are hidden from organizations that do not belong to them. For instance, R1 isn't aware of the existence of C2, despite R2 and R0 being stakeholders of both channels.

The Membership Service Provider (MSP) manages the identities and maps the certificates to individual roles and permissions within the network. The MSP establishes the rules for the authentication and authorization of entities, ensuring that only peers with a valid, CA-signed certificate are allowed to interact with the network according to the access rights defined by the organization's policies.

Orderers are nodes responsible for packaging transactions into blocks and ordering the blocks in a definitive order. The green "O" is the only ordering node represented on the diagram, but the role might be played by multiple nodes. This ordering service is done according to a consensus mechanism, which ensures that transactions are validated and committed in a consistent order across the network (more on this later).

Smart contracts are called "chaincodes" within the Fabric terminology. cc1 is a chancode belonging to C1 on the diagram. S5 is the state of the chaincode cc1 that is run on every node endorsing the smart contract (P1 and P2), but not on ordering nodes (O). A channel, associated with a unique ledger, can contain multiple chaincodes, and each chaincode may have different organization rights.

Client applications (A1, A2, A3) can interact with the smart contracts using Fabric Gateway, a feature that constructs and send transactions to peers using the SDK and API. These clients must be authenticated and have a certificate that proves that they belong to an authorized organization.

In summary, the peers form the backbone of the Fabric network, hosting ledgers and chaincodes, and interacting with client applications and orderers to maintain an updated and consistent ledger. Organizations own these peers. The CAs and MSPs ensure that the entities are authenticated and authorized to perform their roles within the network.

Transactions phases

In typical blockchain architectures, transaction processing is straightfoward: a client generates and signs a transaction and forwards it to a node. This node then propagates the transaction across the network. A miner (or a validator, depending on the consensus algorithm), aggregates several transactions into a block, appends this block to the ledger, and broadcasts it throughout the network. It is worth noting that blocks may be discarded during a fork—a scenario where the majority of miners choose to extend the blockchain with a different block.

In the Hyperledger Fabric framework, the transaction process initiates with client applications generating transaction proposals. These proposals are sent to endorsing nodes, as defined in the endorsement policy, which indicates the list of peers who must endorse transactions interacting with a specified chaincode. These peers simulate the transactions against their local copy of the ledger and return a signed proposal response, termed as the read-write set. At this stage, the ledger is not yet updated. After successful endorsement, the client application assembles the responses from all the endorsing nodes into a transaction and submits it to the ordering nodes. The orderers do not execute or evaluate the transaction's content but simply verify the signature and authorization, then batch transactions into blocks and order them, ensuring no double-spending or conflicting transactions are included within the same block. The fact that orderers do not execute code permits to avoid bottlenecks, as the execution and ordering process are separated into distinct roles, therefore increasing scalability. Once ordered, the block will be distributed to all peers of the channel (by broadcast or via cascading using the gossip protocol). Each peer validates each transaction in the block independently, ensuring that ledgers remain consistent, then commits the block to its local copy of the ledger.

Ordering nodes and consensus algorithm

Up until now, we only said that transactions were ordered by ordering nodes (or orderers) forming the ordering service, but we did not explain the consensus algorithms behind this process. Fabric supported different consensus algorithms: Solo and Kafka which are now deprecated, Raft, and BFT since v3.0.

Contrary to the consensus protocols of Bitcoin (PoW) and Ethereum (PoS) which are probabilists, Fabric relies on deterministic consensus algorithms. Therefore any block validated by the peer is guaranteed to be final and correct, and forks cannot occur.

The **Raft** protocol operates on a "leader and followers" model. It is said to be *"crash fault-tolerant"* (CFT) because it can sustain the loss of nodes, (including the leader), as long as there is a majority of ordering nodes (quorum) remaining. In this model, a leader node is dynamically elected from among the orderers within a channel, collectively referred to as the "consenter set." The consenters periodically elect a leader. This leader is responsible for processing log entries and replicating them to follower nodes. If the leader ceases to send periodic "heartbeat" messages—a signal of active status—the follower nodes are programmed to initiate a new election to select a new leader. The leadership in Raft is dynamic and specific to each channel, allowing for different leaders for different channels in a multi-channel setup.

On the other hand, the **BFT** (*Byzantine Fault Tolerant*) protocol is designed to not only handle node failures but also if up to a third of its nodes are compromised or act maliciously. The name comes from the Byzantine Generals Problem, invented in 1982 by Leslie Lamport and Robert Shostak and published in [90]. The Fabric's BFT orderer implements the SmartBFT protocol, itself derived from the Practical BFT (**PBFT**) [26]. It is also based on a "leader and followers" model, but the difference is that the leader is rotated among the nodes in round-robin manner whenever the current leader is suspected by the followers of malfunctioning. Unlike Raft, when a client initiates a transaction to a BFT ordering service, she must broadcast it to all nodes rather than just one. This way, if the follower nodes receive a transaction from a client, and the leader acting maliciously ignores it, they will eventually replace it. The client can be certain her transaction will be included in a block, as a new (and honest) leader will eventually add the transaction.

In summary, while Raft offers an effective solution for networks with trusted nodes but a risk of failures, BFT extends these guarantees to environments where up to a third of the nodes may act maliciously.

3.4 Description of the current Belgian voting system

This section describes the Belgian voting system as it is for now. The technical specifications may slightly change from one election to another, and the specifications for the next elections are not yet decided at the time of writing. For instance, the minimum legal age for the next election of the European Parliament in Belgium in 2024 should be lowered to 16, but the law of the 1st June 2022 [5] that gave citizens the option of voting from the age of 16 has been revoked by the Constitutional Court [34] who judged that requiring minors voters to register prior to the election was unconstitutional. Because of these uncertainties, we will base on the last two elections that occurred in 2018 and 2019.

Voting right

Elections are made by universal suffrage. Voting is compulsory for all Belgian citizens over the age of 18 who have not been deprived of their right to vote. All voters have one vote. EU citizens residing in Belgium for a minimum of five years are entitled to register and participate in both European and municipal elections. Similarly, non-EU residents who have lived in Belgium for at least five years are eligible to register and vote in municipal elections. The complete list of conditions for the voting rights can be found on the Direction of the Elections website [41]

Rules of voting

The federal election in Belgium is not a first-past-the-post system (i.e. the voter votes for one candidate, and the candidate with the most votes wins) as in many countries. It is instead a multi-party system (i.e multiple parties generally have to make a coalition to govern). Furthermore, there is also a preference voting system. The voter must vote for a single party in the list, but can then select as many of that party's candidates as she wants to give them a *preference vote*. She may also select the *head-of-list*, that is, she gives no preference vote, and accepts the order of the list as proposed by the party. The principle of "one human, one vote" is respected, since each vote brings a single vote to the selected party. Preference votes are only used to favor a candidate within its own party. A voter can only vote for the candidates of her constituency.

Types of elections

For the sake of brevity, we do not go into details about the country's federal structure. We will simply observe that the federated institutions are divided geographically into regions (Flanders, Wallonia, and Brussels-Capital), and culturally into communities (Dutch-speaking, French-speaking, and German-speaking). To avoid having too many election days and confusing the public, several different elections can take place on the same day. Here is the list of different types of elections, with the date of the last and next and occurrence at the time of writing:

- Municipal elections, the most local elections (28/10/2018 - 13/10/2024)
- Provincial elections (28/10/2018 - 13/10/2024)
- Region and communities' elections (26/05/2019 - 09/06/2024)
- European Parliament's elections (26/05/2019 - 09/06/2024)
- Federal elections for the House of Representatives (26/05/2019 - 09/06/2024)

As discussed in the introduction, referendums are not legal in Belgium. Some "popular consultation" can be proposed, but only occurrence goes back to 1950. Naturally, back then the vote was done traditionally and not electronically.

Election's process

For the 2018 and 2019 elections, the electronic voting system was applied in 157 municipalities of Flanders, the 19 municipalities of Brussels-Capital, and the 9 German-speaking municipalities of Wallonia. The rest of Wallonia and the other flemish municipalities opted for a traditional paper-based voting system.

According to [45], the process of the electronic e-voting election is as follows: Before the election day, the convocations to vote are sent to all the citizens with the right to vote. The president and the attendants of the polling station are nominated and receive a short training. On election day, the president of the polling station turns on the machines and opens the station to the public. During the election, the elector first gives her ID card and her convocation to prove that she is allowed to vote. She then receives a magnetic card to vote. The card is only used to start the electronic process, and does not contain any information on the user, nor is used to register the vote. The user goes to the voting booth and uses their card to start the process. On a digital screen, the user proceeds to the actual vote in the language of her choice, among the official languages of the location. First, the elector selects her favorite party among the list of parties or chooses to blank vote. After the confirmation, if she selected a party, she then gives a preference vote to as many candidates of the selected party as she wants, or selects the "head of the list" if the elector has no preference for any candidate. Once the user has confirmed her choice, her vote is final and cannot be modified, canceled, or restarted. The process is repeated for every election happening on the same day.
After the confirmation of her vote, the machine prints the ballot on a paper with a barcode, and the content of the vote is also written textually so that the elector can be sure of what was taken into account. Optionally, the elector can go to another voting

booth to check if the content of the barcode corresponds to what is written on the paper. If this was not the case, then it would mean that the result of the electronic voting could be invalidated. To validate her vote, the elector goes to the ballot box and lets the machine scan the barcode. Once it is done, a slot opens and the elector can drop her paper inside the electronic ballot box machine. The vote is now registered in the machine and paper is kept as proof (e.g. in case of problems, to recount the ballot). The elector can then return its magnetic card to retrieve its ID card and leave the polling station. At the end of the day, the results are sent to the central authority by a software called Martine. When all the results are collected, the results are officially published.

Electronic voting machine

The electronic voting machines are developed by the private company Smartmatic. The company explains on its website how the machines work. Three types of machines are used in the election process: the voting machines, the President Machine that manages the polling station, and the e-Urn (electronic ballot box).
The **Voting machines**, in the voting booth, have a touchscreen to let the user proceed to vote and a built-in printer to produce paper ballots (with a machine-readable barcode and a human-readable text).
The **President Machine** has four main functions: To activate magnetic cards that voters use to access the voting machines; to electronically register and store each vote; to count all votes and store the results; and to generate a polling station report.
The **e-Urn** scans the ballot's barcode and stores the paper ballot.
Smartmatic indicates having deployed in 2019 for the election 22.850 voting machines, 4338 President Machines, 4338 e-Urn, 14.000 USB keys, and 235.000 magnetic cards.

The software used for collecting data of candidates, gathering the results, and transmitting them to the authorities is called **MARTINE** (Management, Registration and Transmission of INformation and results about Election) and was developed by a Belgian private company Civadis. As required by the law, parts of the source code of the software have been released the week after the election.

Historic of the Belgian voting system

Belgium's electoral system has evolved significantly since its inception. At the creation of the kingdom after its independence in 1831, it began with census suffrage, limiting voting to male citizens over 25 who paid certain taxes. By 1893, general multiple voting rights were introduced, expanding eligibility and allowing additional votes based on education or tax payments. Following World War I, in 1919, the general single suffrage lowered the voting age to 21 and granted each man one vote. Finally, after World War II, women were granted voting rights in 1948, achieving universal suffrage.

Before 1991, all votes were cast in the old traditional way using paper. In 1991, the first pilots of an electronic vote were launched, and during the 1994 election, 22% of the population voted electronically [136]. At that time, the electronic vote was registered on a magnetic card and the elector had no way to verify that her vote was well taken into account. Despite the population seeing electronic voting as a positive initiative, the impossibility of verifying the result led to a loss of trust. For this reason, the next model was designed to be also human-readable: the elector received a paper with the content of its vote, and this paper is kept in a ballot box along with the machines-vote database.

In the late 1990s and early 2000, another electronic system was tested, where paper where cast on paper but electronically tallied using an optic scanner [125]. In the 2003 elections, an electronic voting glitch resulted in a candidate receiving 4,096 extra votes. It led to an impossible scenario where the candidate had more votes than her list. The issue was attributed to a memory error in the voting computer, potentially due to a cosmic ray causing a flip of a bit [101].

In 2020, the Federal Public Service of the Interior commissioned a study [132] that studied the possibility of voting through the Internet during the Belgian federal election. The study concludes that voting through the internet in Belgium could be possible from 2034, but that a pilot test could be done during the 2024 election. However, at the time of writing, no budget has been allocated to this initiative, making it very unlikely.

3.5 Available digital tools

Electronic ID card

Belgian *electronic identity card (eID)* are equipped with a chip that can be used for authentication and electronic signature. The chip contains an electronic copy of the citizen's information (name, gender, national security number, address, card number, validity dates, JPEG ID photo, ...). The chip also contains 2 secret keys (one for authentication, one for signature) and their associated public keys, along with their respective certificates signed by the certification authority (Belgian government). The chip, equipped with a small CPU, can compute cryptographic calculations [33]. Belgians' IDs are ISO 7816-compliant, which makes them readable from most card-readers, and electronic signatures have the same legal value as a handwritten signature.

Since 2021, the newly emitted eID cards contain an electronic copy of the owner's fingerprint. However, these are not readable by regular card-reader and can only be accessed by competent federal authorities (such as the police). According to the law, these fingerprints will be deleted from the SPF's database three months after their registration. Therefore, there are no databases containing the fingerprints of all citizens.

Itsme®

Also, a mobile application itsme® provides Qualified Electronic Signatures after a pre-configuration using the eID and a card reader or bank identification. Itsme® is compliant with the eIDAS regulation [36] (European regulation on electronic identification and trust services for electronic transactions [48]). This application is part of the CSAM offer, a set of authentication methods allowing to log into online public services.

European Digital Identity Wallet

Under the initiative of the European Commission, a pilot experiment for a European Digital Identity Wallet is currently being developed. Expected to be launched for the general public in 2025, this wallet will enable any European citizen to identify with public and private digital services across the EU. With a focus on privacy, users should have full control over the data they share and decide exactly what data will be shared at any authentication. Some use cases of this project are for instance providing secure access to various services, including government services, opening bank accounts, SIM registration, mobile driving licenses, contract signing, prescription claims, and travel document presentation. Additionally, it serves as proof of identity for educational certifications, accessing social security benefits, and facilitating freedom of movement. Currently, four large-scale cross-border pilots are developed in parallel. The pilot focusing on secure qualified digital signatures is called Potential [1] and is co-lead by France and Germany, with stakeholders from public institutions, industry, and academics from 19 countries.

EBSI

EBSI (European Blockchain Services Infrastructure) is a pan-European blockchain infrastructure that provides a range of blockchain-based services to public and private organizations. It is an initiative of the European Commission.

EBSI offers permissioned and private blockchains secured by a set of approved nodes, permitting a scalable decentralized network. It is designed to be secure, interoperable, and resilient providing a high level of protection against cyber threats and data breaches. It can be used to build a wide range of blockchain-based applications and services.

The European Digital Identity project and EBSI go hand-in-hand, as some applications are closely related. For instance, another pilot project for the European Digital Identity Wallet is DC4EU. DC4EU aims to provide a service that allows users to obtain credentials generated by an issuer, and let a verifier verify the validity of the credentials. This technology is based on EBSI to guarantee integrity, privacy (control of the data by the holder), verifiability, and availability. For instance, a student could receive a diploma from a university in Europe. This university could issue an electronic version of the diploma stored on the EBSI blockchain and the student could later provide this as proof for a potential employer or another university to continue her studies. The employer plays here the role of verifier. Based on the credentials provided by the student, she can verify by herself on the blockchain the validity of the diploma issued by the university.

In the paper *Verification of Education Credentials on European Blockchain Services Infrastructure (EBSI): Action Research in a Cross-Border Use Case between Belgium and Italy*[127] by E. Tan *et al.*, the authors experiment with a concrete instance of this example. They validate the verification model and highlight the challenge for the wider adoption of the use case.

Chapter 4

Requirements analysis

4.1 Requirements of an election

This section outlines a formal definition of election requirements, both physical and online, followed by an identification of the specific requirements for Belgian elections. It examines how well traditional paper voting and existing electronic and internet voting systems meet these requirements. We will see that some requirements oppose each other in the case of online voting and that we need to balance these requirements to get an ideal model that fulfills our needs. After highlighting the risks and limitations of online voting, we will search to what extent blockchain technology could solve some of the issues, and analyze how a blockchain-based e-voting system adapts to the requirements of an election. Finally, these requirements will be used in the next section to establish a formal methodology for identifying the optimal blockchain-based voting system that meets them all.

Many papers have formalized the requirements of an election, especially for online elections, but the terminologies vary. We will divide our requirements into distinct categories, to follow the methodology of Qadah and Taha in *Electronic voting systems: Requirements, design, and implementation*[113] who divide the requirements in three categories: generic, system-specific, and election-specific. We will use the same categories, but divide system-specific requirements into e-voting-specific, and i-voting-specific requirements. In Section 4.2, we will compare how different voting systems apply those requirements.

4.1.1 Generic requirements

This category includes all requirements that are generic to any voting system (including paper-based).

- **Integrity/Tamper-proof**: Ballots and results cannot be altered by any third party. The registered vote must correspond to the real voter's choice. All the votes must be taken into account, no vote can be removed or modified, and no "fake vote" can be added.
- **Privacy/Anonymity :** The system must guarantee the anonymity and the privacy of the voters. A vote can't be linked to the voter's identity (Anonymity), and a voter can't be linked to her vote (Privacy). This requirement is sometimes called *Untraceability* or *Unlinkability*.
- **Authenticity/Eligibility :** Only people that are allowed to vote can vote.
- **One human, one vote :** Each voter can dispose of only one vote. There is an exception in the case of plurinominal balloting (each voter can vote for multiple candidates), but in that case, one vote containing the set of selected candidates is sent per voter. Also, in the case of a voting proxy, an individual can vote several times because he or she represents several voters. This requirement is sometimes called *Unreusability*, as the voter cannot use his or her right to vote multiple times.
- **Availability/Timing :** All the votes must be expressed in a determined interval of time. After the final limit, no more votes must be accepted. Also, all voters should have the possibility to vote during the voting period.

- **Auditability/Verifiability :** All the processes must be audited before the election to guarantee security and fairness, and all votes must be re-countable by external auditors after the election. During the election, observers make sure that no vote manipulation can occur, and that all votes are taken into account.
- **Freeness :** Every voter can vote for free.
- **Security :** The system must be resilient to any type of fraud or attack (including physical access) throughout the voting process, to guarantee the respect of the aforementioned requirements. The security aspects depend largely on the voting system.

4.1.2 e-voting spefic requirements

In electronic voting systems, *i.e.* where voting is done via electronic machines at polling stations, all previously mentioned requirements must be adhered to. In addition, these systems bring into play new specific requirements that must also be met.

- **Hardware and software security** must be ensured to avoid attacks that would result in the violation of one or more of the requirements aforementioned. This includes the impossibility of inserting malware even with physical access to the voting machines inside the voting booth, the protection of the software of the voting machines, and protection of the tallying, and recording transmission of the results.
- **Logs** must be stored to allow later audit, but not contain sensitive data that could jeopardize the voter's privacy. These logs must be tamper-proof to preserve integrity. In [9] Alamleh and Alqahtani suggest the use of blockchain to protect the integrity of logs.
- **Open Source :** For the sake of transparency, all the codes should be Open Source, without compromising security. Also, the compiled code of the software and voting machine must match the published code.
- **Transparency**: The voter must be certain that her vote is well cast and registered, i.e. it corresponds to the candidate(s) she selected.
- **Accessibility :** The voting process must be designed to be easily usable by all eligible voters, including those with disabilities. It can be achieved through font size, language, adaptation for blind and color-blind people, etc. These are *non-functional requirements*, but remain important for inclusiveness.

4.1.3 i-voting specific requirements

Remote, or Internet-voting (i-voting) has also specific requirements in addition to all of the above.

- One might think that a good feature would be that a voter can check that her vote is well taken into account and corresponds to what she voted for. However, this functionality has a downside: a coercer could try to force a voter to prove whom she voted for. Moreover, in the case of a steal or a leak of its private key, her vote would be known. An adjective that describes the impossibility of having a receipt of a vote is called *receipt-freeness*.
- **Receipt-freeness :** The elector does not receive a receipt of its vote. She then cannot prove to anyone the content of his/her vote. This property is largely discussed in the literature. J. Benaloh and D. Tuinstra [20] justify it by «*avoiding vote buy and coercion*».
- **Coercion-resistance :** The voters can make their choices freely and privately, without any external pressure or influence that could compromise their vote. Coercion also includes vote buying.
- **Hybrid internet-paper vote :** It might be difficult to imagine for a lot of reasons to go from a system with no possibility to vote online, to a system 100% online. A transition between the two states could be giving citizens the choice to vote either on-site or online, but not both. That way, people not comfortable with computers would not feel lost. In such a model, a voter opting for online

voting would be registered as having voted electronically, maintaining anonymity regarding their specific choice, to preserve vote secrecy.

4.1.4 Online voting conflicting requirements

Online voting presents certain challenges in achieving all the properties of a vote that traditional paper-based voting systems offer. Indeed, some properties might be contradictory with each other.

- **Verification vs. Non-coercion**: as explained in the previous section, voters may want to verify that their vote has been registered as expected. But this might alter the non-coercion requirements (as a voter could prove for whom he/she voted for and therefore someone could influence their vote). It would also represent a challenge to privacy, as there normally shouldn't be any possibility to establish a link between a voter and its vote. In [55], T. Finogina and J. Herranz propose a model of remote electronic voting that achieves both coercion resistance and cast-as-intended verifiability, but at the cost of other requirements not being respected.
- **Anonymity vs. authentication:** it is a significant challenge to have a robust authentication method to verify a voter's identity to prevent fraud, while not affecting their anonymity.
- **Scalability vs. Security**: Achieving an exceptionally high level of security while concurrently managing a substantial volume of votes represents a challenging equilibrium.
- **Transparency vs. Ballot Secrecy**: The system must ensure transparency. However, revealing too much information may compromise ballot secrecy.

4.1.5 Election-specific requirements

Election-specific requirements cover the specifity to a particular election compared to generic election. In the case of Belgians elections, we can count at least 5 additional requirements.

- **Mandatory vote:** In some countries (such as in Belgium), the vote is mandatory. Therefore, the system must remember if an elector has proceeded to vote.
- **Blank** votes are a specific type of vote where an elector votes for no candidates or no party. It is different from null votes (i.e invalid votes). Voters can blank vote for different reasons, for example, if the vote is mandatory but abstention is allowed with a blank vote, or to express their disagreement with all candidates. Under Belgian law, blank and invalid votes are counted but are not taken into account when allocating seats between parties [119].
- **Multiple elections:** Belgium may host multiple elections on the same day (see Section 3.4). Therefore, an elector should be able to vote for the different elections in a row, without reiterating a long process, but also without compromising security (e.g. by reusing a one-time key).
- **Preference vote:** Belgian electoral system is not a first-past-the-post system, but a system based on parties and preference vote (see section 3.4). The voter selected one party and, optionally, one or more candidates within this party. This choice is cast as one vote.
- **ID verifiability :** The voters must prove their identity using their ID card. This could also be theoretically achieved electronically using the eID and a card reader (see Section 3.5), using the mobile app itsme (section 3.5), or in the near future with the European Digital ID (Section 3.5). Nevertheless, the requirement of anonymity requires that no link can be established between the elector's identity and the content of a vote.

4.1.6 Optional features

Some voting properties could be interesting to have, but are not (at least according to current voting laws) mandatory.

- **Modify a vote:** With online voting, some models let the voter cast a vote multiple times in the case the elector changes her mind, and only the last vote is counted. Some consider this property to be a remedy against vote coercion, as a voter could prove for whom she voted, but then modify her vote [91]. However, this statement does not make unanimity in the literature [55]. Note that this property is not allowed yet in Belgian law, but if online voting was ever to be implemented, this could become a design choice.
- **Flexibility:** A desirable property for the model would be to be flexible for other kinds of elections (i.e. the model could be used for Belgian elections, but also for referendum, or be used in other countries' elections).

4.2 Comparison of voting systems

In this section, we will analyze, for each type of voting process we consider in this thesis (paper, e-voting, and i-voting) how are each requirement respected (or not), as well as the advantages and drawbacks compared to each other.

4.2.1 Analysis of traditional paper-based voting

The paper-based voting system is the traditional voting process that is used in most of the countries in the world. In this system, a registered elector puts a paper with the name of a candidate or a party in a ballot box. At the end of the voting period, the ballot box is opened and the votes are counted. A more detailed description of the entire voting process in a traditional voting system has been written by Dimitris A. Gritzalis in *Principles and requirements for a secure e-voting system*[65]. Let's see how are each requirement of section 4.1 respected.

- Authenticity/Eligibility: eligible voters are registered beforehand and either receive an admission card or are registered on a list in their attributed voting station. Only registered voters have access to the voting booth.
- Privacy/Anonymity: The vote is done secretly in the voting booth. The paper is folded on itself or placed inside an envelope so that other people cannot identify the content of the vote. Once placed inside the ballot box, all votes are identical and no link between a voter and a vote can be established.
- Integrity/Tamper-proof: The ballot box cannot be opened during the voting period. Once a vote is inserted, it cannot be altered or removed (neither by the voter itself nor by anyone else). As the ballot box is usually transparent and highly visible, it is very unlikely that someone could add an authorized vote, or remove or change the votes inside of it. At the end of the election, the ballot box is sealed, to protect the integrity of the votes.
- One human, one vote: A voter can only place one paper in the ballot box. If a voter has voted for two names (e.g. two papers in an envelope, or two different parties ticked in a list), the vote is accounted as a "null vote". Once a voter has voted, it cannot reproduce the same process another time (e.g. she doesn't have her voting card anymore, or her name has been removed from the list. In some countries, the voter's finger is dipped in indelible ink). In Belgium's preference vote system, a vote with multiple candidates selected belonging to the same party is a valid vote.
- Availability/Timing: The voting poll is accessible during the entire voting time, and is closed at the time of election closing. Depending on the country, workers sometimes have a disposition to go to a voting station on election day. For citizens not able to come on-site, proxy votes are authorized.
- Free of charge: Every voter votes for free.
- Security: safety relies on compliance with procedures. The security is assured by election observers which are people from different parties and citizens selected randomly among the population living in the district of the voting station that are there to guarantee the respect of the process. Police agents might be there to protect the voting station.

- Auditability: the process is verified by the election observers cited above. Any suspicion of fraud is reviewed. External and independent auditors might verify if the procedures have been respected.

Pro and Cons of paper-based voting systems

PROs:

- The traditional paper voting respects all the requirements of an election listed in Section 4.1.
- Secure, reliable, and widely used for long in many countries.
- Easy to implement, as it does not rely on high-tech machines complicated to set up.
- Easy (but time-consuming) to tally the votes.
- Easy to vote, even for non-tech savvy.
- Acceptance by the public: People have trust in the system and understand the process. They know that their vote has been cast and recorded as intended.
- Multiple trustees to avoid "one person of failure".

CONs :

- Expensive at each election (paper, people).
- Complex logistics to install the voting stations.
- The tally takes time, as a consequence the results are not always directly published.
- Requires every voter to vote in its assigned district. Some voters are unable to get to the polling station to vote due to many reasons (illness, being abroad, ...)
- Relies on trust in election officials and legal procedures.
- Not publicly verifiable, and therefore not 100% safe against state-level manipulation (e.g. in not-democratic countries, but also in western democracies like the suspected attempted electoral fraud in Georgia during the 2020 US election, or the seized ballot boxes by the Spanish Federal government during the 2017 Catalonia's Independence referendum [46]).

4.2.2 Analysis of e-voting system

In an electronic voting system (or e-voting), the voters proceed to vote on machines located in the polling stations. The process to access the voting booth is the same as for paper voting, so the requirements of eligibility, availability, and freeness are guaranteed the same way, as described in the previous section. However, electronic voting brings new challenges.

- Privacy/Anonymity: The vote is done secretly in the voting booth. However, the software and hardware design must guarantee that no link between the voter and her vote can be established.
- Integrity/Tamper-proof: The software and hardware design must guarantee that all votes are properly registered and cannot be altered.
- One human, one vote: The machine can automatically detect that a vote is invalid, and notify the voter. To guarantee that an individual cannot enter the voting booth multiple times, the same approach as traditional voting is employed. Both the software and hardware must be designed to prevent a voter from fraudulently casting multiple votes simultaneously
- Auditability: Additionally to the audit of the traditional vote, the machines and the code of the software must also be audited, and the logs can be used to verify the results and detect fraud.
- Security: the security of the hardware and the software represents an important threat. Security of the machines must be guaranteed in order to satisfy the requirement of privacy, integrity, and fraud protection.

The risks with e-voting

The use of electronic machines brings new threats to voting systems. As for any computer device, cybersecurity is a severe outcome, as it may be subject to bugs or attacks. Moreover, such a high stake as an election might be targeted by multiple adversaries. According to Halderman in [72], possible threats may arise from system administrators, cybercriminals employed by dishonest candidates, activists with hacking expertise seeking to disrupt elections as a form of political protest, and even advanced nation-states employing offensive cyberwarfare capabilities. Halderman divides these attackers' goals into three categories: (1) Tampering with the election outcome (attacking the *integrity* property) to e.g. favor a specific candidate; (2) Discovering how people voted (attacking the *anonymity* property) e.g. as a means of enforcing vote buying or coercion; and (3) Disrupting or discrediting the election process, e.g., through denial of service, obvious tampering, or the false appearance of such problems (attacking the *availability* property and *security* in general, and overall the *trust* in the election). Also, defending all these aspects simultaneously is challenging. For instance, a usual way to protect a system is through the use of logs. Logging data (such as actions performed by users, connections, etc) is a good practice to recover in case of attacks. But in the case of an election, extensive logging might put at risk the privacy of the user. For instance, logging the timestamp, the machine ID, and the content of a vote will reveal the vote of the person who entered that specific voting booth at that specific timestamp [72]. Halderman also argues that it is almost impossible to guarantee that the code is without bug and that it must be proven that it is the official code (and only this one) that is running on the voting machine.

4.2.3 e-voting in reality

Different types of e-voting were proposed and have been used around the world. Most of the models can be classified into two sub-categories of e-voting: Those that use direct-recording electronic voting machines (DRE), and the hybrid systems that use voter-verifiable paper audit trail (VVPAT). In some earlier voting systems, traditional paper ballots were used alongside electronic counting methods. These systems employed optical scanners [125] to read the paper ballots or utilized punched cards [3] for vote recording. This approach combined physical voting mediums with digital tallying technology. However, those will not be discussed in this paper, to focus only on DRE and VVPAT.

DRE

With direct-recording electronic voting machines (DRE), votes are directly processed and stored in the system. The user cannot verify if the vote is cast as expected. The requirements of anonymity, integrity, one human one vote, auditability, and security depend entirely on the design of the hardware and software of the voting machines. Since their creation, they have met many criticisms and have evolved. Gibson *et al.* considers in *A review of E-voting: the past, present and future* that *«the vast majority of experts would now acknowledge that—even though there are many reported and ongoing problems with systems that are currently in use - such systems can be built and operated in a satisfactory manner provided they support some form of voter-verified printed audit trail (VVPAT) »*[61].

According to Halderman, DRE *are essentially general-purpose computers running specialized election software* [72]. It is therefore hard to guarantee a good level of security (as general-purpose computers are themselves often victims of cyberattacks). Also, electronic voting machines are built to last 20 to 30 years. This means that not only do these machines need to be secure now, but also require to remain secured in a few decades [72].

The greatest example of such machine failure is the Diebold AccuVote-TS voting machines, a type of DRE widely deployed in the United States and used for general elections in the mid-2000s. An analysis from A.J. Feldman, J.A. Halderman, and E.W.

Felten has put evidence on the vulnerabilities of these machines to extremely serious attacks. They have shown that an attacker who gets physical access to a voting machine for only one minute could install malicious code to modify the votes and logs on the machine, and spread a virus to contaminate other voting machines [54].

In conclusion, direct-recording electronic voting machines (DRE) suffer from a lack of transparency. The authenticity, integrity, and anonymity depend entirely on the software and hardware designs and rely on the machine's developers and the experts that control the election. In the past, such machines used in actual elections have seen their security later broken. The impossibility for the elector to verify their vote and to understand the underlying process has resulted in a lack of trust by the general public. For these reasons, these systems have often been replaced by VVPAT. This is also the case in Belgium, where the first generation of electronic voting machines have been replaced by a new system with VVPAT in 2012 [136]. In Germany, the Federal Constitutional Court ruled in 2009 that the use of DRE for the 2005 election was unconstitutional and that a citizen without specialist knowledge should be able to vote and determine the results[4, 125].

Voter verified printed audit trail (VVPAT)

In this model, voters utilize electronic voting machines to cast their votes, and these machines generate a printed paper record indicating the voter's choice in a format that is both comprehensible to humans and interpretable by machines. The system includes a mechanism for verifying that the code recorded by the machine aligns with the human-readable information. Votes are initially counted electronically, but the printed records are securely stored in the ballot box to facilitate potential recounts. [136, 61]. Villafiorita *et al.* provided in [137] a detailed description of this system and the use of such machines in Italy. The property of verifiability in this electronic voting system is achieved by the presence of human-readable paper records. It ensures the protection of authenticity and integrity: in instances of potential electronic counting errors, a paper-based recount option remains available. Assuming voters verify that the human-readable text aligns with their choices, the content of the ballot box maintains its integrity. Nevertheless, the electronic count, while more secure, is not without minimal risk, as a recount usually occurs only in cases of doubt, not by default.

Regarding anonymity, the paper-based aspect of the system provides the same level of privacy as traditional paper-based voting. However, the process of casting the voting on an electronic machine remains the same as for DRE. It is crucial to avoid establishing any connections that could compromise anonymity (e.g. using anonymous magnetic cards to activate voting machines, not intrusive logs, etc). Regarding auditability, the system offers the flexibility of auditing every phase before, during, and after the election. In the event of a recount, it becomes a straightforward process to verify that the results obtained from the paper records align with the electronically counted outcomes. In terms of security, the system is notably more secure than Direct Recording Electronic (DRE) systems, primarily because it guarantees the integrity and authenticity of the paper ballot box. While electronic voting machines and electronic counting devices remain vulnerable to potential security risks, any issues that may arise can be detected and solved with a recount of the papers in the ballot box.

In essence, electronic vote counting closely follows the process of traditional paper-based voting. Consequently, e-voting systems employing Voter-Verified Paper Audit Trails (VVPAT) tend to induce greater public trust for these aforementioned reasons.

Pro and Cons of e-voting system

PROs:

- Similar process as for paper voting to access the voting booth, ensuring eligibility.
- Automatic tallying of votes, and collation of results [39], resulting in almost immediate obtention of the results.
- Elimination of human error and bias in the recording of valid votes [39] and in the counting process.

- In the case of VVPAT, *cast-as-intended verifiability* is ensured by the human-readable text printed on the paper.

CONs:
- High cost (most of the cost of traditional elections, plus the purchase and maintenance of electronic machines and devices).
- Even more complex logistics.
- Like for paper voting, electors must vote in their assigned district.
- Accessibility for non-tech-savvy people (digital division).
- Risk of violation of vote secrecy or anonymity (especially in the case of DRE).
- Risk of malware, bugs, and cyberattacks (especially in the case of DRE).
- The poll workers are not professionals. They are selected arbitrarily among the citizens of the municipality a few days before the election and receive very short training. They may not be familiar with the technology and are therefore likely to make a mistake (a new source of human error), or in the worst case to be subject to corruption.
- The collation of results relies on centralized system architectures, which make them susceptible to cyberattacks that target central infrastructures [39].
- The results are seen by only a small amount of people among the election authority before being officially published, instead of being publicly readable.
- Many e-voting software are proprietary and protected by trade secrets instead of being open source. As a result, only a tiny number of experts can audit the security of the code [125].
- The system is based on trust in people (poll workers, witnesses, software developers, machine constructors, auditors, ...) and on laws preventing fraud, instead of being *de facto* secure.

Problems with current system in Belgium

As described in section 3.4, Belgium uses simultaneously an e-voting similar to the e-voting with VVTAP explained above mainly in Brussels and Flanders, and a paper voting system in the other municipalities. The e-voting part is the focus of attention. A description of the hardware and software used in this section is provided in Section 3.4. Based on the report of experts that analyzed the IT security of the 2019 election [45], multiple issues and malfunctions have been raised.

- The Smartmatic machines and the Martine software are two single-point-of-failure. A small bug somewhere could potentially compromise the entire election.
- Moreover, these systems are developed by two distinct private companies. According to the report [45], these two companies blame each other for the lack of compatibility. The system is audited by another private company (PwC).
- The group of experts sent by the authorities to check the compliance complained of a lack of time and resources to complete their mission.
- The law regulating code publication is unclear about which parts of code must be open source.
- The report [45] lists many problems that were observed during the last elections, including **flawed material, non-compliance with procedures, and human errors**.
- For instance, in the 2018 Brussels election, an electronic counting issue at a polling station recorded only 58 out of 885 votes cast. This discrepancy was traced back to a software glitch in the counting system. Fortunately, since the paper ballots were securely stored in a sealed ballot box, a manual recount was conducted two weeks later to rectify the error [49].
- The **cost** of an election is important: thousands of voting stations, a huge amount of hardware, a lot of paid workers, and a high cost of software development. According to an external source [80], the development of the Martine software alone has cost 420.000€.

4.2.4 Analysis of i-voting system

In remote voting, voters can proceed to vote from home, without the need to move to a voting station. Chantal Enguehard analyzed the vulnerabilities of three distinct remote voting systems: namely online voting, postal voting, and a hybrid system combining postal voting with electronic counting, as outlined in [50]. Furthermore, Gibson *et al.* also undertook an analysis of postal voting, detailed in [61], while some papers have explored alternative methods for remote voting, such as voting via SMS [42] or email [93]. This paper, however, exclusively concentrates on online remote voting over the internet. Therefore, the terms "remote voting", "online voting", and "i-voting" are used interchangeably to refer to the same concept. In online voting, eligible voters can submit their ballots from any location, using an electronic device connected to the Internet. These votes are usually transmitted to a central server that collects all the votes and tallies the results.

One key question in online voting is how to safeguard the **integrity** of these votes when they are stored on a central server. Even if this server is hypothetically secure against any attack, the entire trust in the process relies on the government and election officials, with no means for an average citizen to independently verify the results. Independent **auditing** is the only method of verification, but if the entire process is managed by the outgoing government, the potential for fraud remains a concern. Unlike traditional polling stations where each station independently tallies votes before transmitting their counts to a central server, online voting relies only on this central entity, thus losing the shared responsibility provided by election watchers and station presidents.

To ensure **authenticity**, citizens must prove their identity on their devices. This often necessitates the use of online authentication tools, which must be developed and deployed prior to the election. This authentication tool itself becomes a central component of the entire model's security framework. However, once a voter is authenticated to cast their vote, ensuring **anonymity** during the voting process becomes a critical challenge. Balancing between anonymity and authentication is one of the challenges of online voting. How can a voter be confident in the real anonymity of their vote? The system's design should unequivocally prevent any potential association between the voter's identity and the content of their vote. Additionally, encrypting the vote is imperative when transmitting it over the public internet to mitigate the risk of eavesdropping.

The principle of **"one human, one vote"** also depends on the system's ability to ensure that a voter can only cast one vote (or the appropriate number of votes in the case of a proxy vote). Gupta *et al.* consider that one risk of i-voting is that a voter could manage to vote multiple times [93]. Concerning **accessibility**, it is essential to accommodate non-tech-savvy individuals with a user-friendly process and interface. Citizens who don't have access to smartphones or computers and/or an internet connection should be offered an option of casting votes at polling stations, similar to traditional elections. To maximize accessibility, the system should work on smartphones and computers across different browsers. Yet, a trade-off between accessibility and security arises as the complexity of developing multi-platform software is higher.

One design choice in i-voting systems concerns whether users should have the ability to **verify** that their votes have been cast as intended. Some researchers, such as in [20], consider receipt-freeness as an effective approach to prevent vote buying and coercion. Some models (such as Helios [6] or the Estonian i-voting system [123]) let an elector cast multiple votes and only count the last emitted vote, so that a coerced voter would have the ability to modify its choice. The debate surrounding the balance between verification and non-coercion has already been discussed in Section 4.1.3.

The risks with i-voting

I-voting contains many new **security** risks and threats. We already discussed the aspect of the centralization of the vote collection. We considered that it was a dangerous design choice if there was doubt in the election's officials, even with the hypothesis that such a server was perfectly secured. However, such a level of security is impossible to guarantee, and therefore the central database containing all the votes is itself at risk of cyberattacks, threatening even more the integrity, authentication, anonymity, eligibility, and one human, one vote properties. We also discussed how the authentication tool is a central piece in the process of security and might represent a single point of failure.

To these risks, we can add all security risks related to the use of the Internet. As the votes are done remotely, an attacker does not need to get physical access to the voting poll, and therefore an attack can theoretically come from anywhere in the world. This puts at risk of a massive cyberattack proceeded or sponsored by an adversarial nation. A DDoS attack would block the availability of the server. An equilibrium between guaranteeing a sufficient level of security while scaling enough must be found. The entire public internet infrastructure must be considered unsecured. A good practice would be to use the Dolev-Yao attacker model, in which it is assumed that the attacker has complete access over the network and can perform any active attack, such as man-in-the-middle, but cannot break cryptographic primitives. Each client device is at risk of a virus or malware. There is also a risk of phishing, redirecting to fraudulent websites to steal the credentials of the users, to detect their vote, to prevent them from actually casting a vote, or all at once.

Academic research in remote voting

An important number of models of i-voting systems have been proposed in the literature (such as [81] [118] [89]). The authors always pretend their model to be highly secure and to prevent fraud. However, the number of countries currently using i-voting in official elections is still very low (see next section). Indeed, despite the promises of supposedly ultra-secure models, Internet voting as a whole is at risk from new threats that cannot be completely avoided.

The study of Enguehard on remote voting system [50] concludes that *"The study notes that the automation of processing combined with the dematerialization of voting objects tends to replace small-scale, visible vulnerabilities with large-scale, invisible ones"*. According to [106], online voting systems are susceptible to significant risks, involving larger-scale, less detectable, and easier to execute attacks compared to traditional paper-ballot-based systems. These vulnerabilities are expected to persist due to the current state of computer security and the high stakes in political elections. The authors consider that introducing such substantial risks into our election systems is an unacceptable trade-off for the convenience of mobile voting.

Analyzing the possibility of using online election in Canada [51], Aleksander Essex recommends not to proceed with Internet voting in federal-level elections until effective end-to-end election verification technologies is guaranteed and a national security framework on Internet voting is established. [44] also argue for complete end-to-end verifiability for public election. Heiberg *et al.* studied in [74] the security risk of voting online from a mobile device. It concludes that smartphones (both Android and iOS) do not offer the same security level as their PC counterpart, especially with the wide range of out-of-date OS versions. The study considers both mobile web browsers and standalone applications. Small screen sizes also bring new problems, such as UI not displaying the entire list of candidates all at once, or the URL bar not showing the TLS connection and complete URL website, which brings risk to phishing to fraudulent websites. A third risk is that mobile devices are often used for two-factor authentication, which is less effective when they are the main device, and they cannot be connected to most card readers.

A commonly highlighted advantage of online voting is its potential to streamline

the voting process for individuals who face challenges in reaching polling stations, such as those living far away, being homebound, or working on election day. Additionally, it is suggested that online voting might stimulate the interest of younger voters, thereby potentially boosting overall election participation. However, the effect of online voting on voter turnout remains a topic of debate in academic research. Some studies argue that electronic voting does not significantly impact voter turnout, such as [135] by K. Vassil *et al.* who says that e-voting «*has not significantly increased voter turnout*», and only had a limited impact on citizens living in peripheral areas. In the paper *From bad to worse: from Internet voting to blockchain voting* [106] by Sunoo Park *et al.*, the authors argue that online voting does not influence voter turnout, referencing multiple sources in the literature. However, the paper incorrectly states that online voting had a negative effect in turnout in Belgium, citing another source [38] to support this claim. Contrary to this assertion, Belgium has never implemented online voting for its elections. The referenced source actually discusses the lack of impact on turnout following the implementation of Direct Recording Electronic (DRE) systems, which relates to electronic voting, not online voting. This misinterpretation raises questions about the validity of the conclusions of the paper. On the other hand, the paper [92] by Madise *et al.* analyzed the impact of internet voting in Estonia over six elections and concluded «*a slight increase in overall turnout*». It also notes that this increase in turnout did not favor specific demographic populations.

To compare the cost of an internet-based election against a traditional one, R. Krimmer *et al.* proposed a methodology for calculating the cost-efficiency of different ways of voting [88]. They concluded that an internet-based voting system was the most cost-efficient.

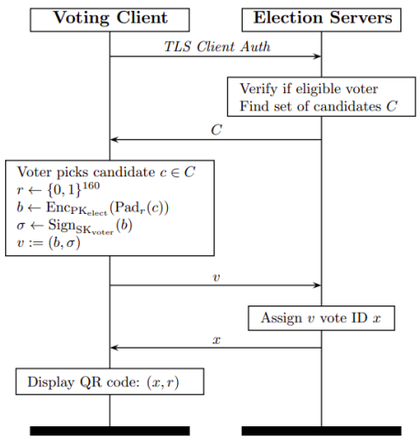
4.2.5 i-voting in reality

Estonian I-voting system

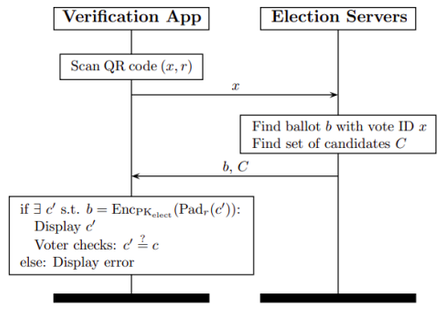
Estonia was the first country to use Internet voting for general elections. The first election using i-voting was conducted in 2005. This system is still in use today, almost 20 years later. This is a hybrid system, where electors have the choice between voting on-site or online. The fraction of remote vote has increased (almost) continuously since then [47], reaching 51% in 2023's Riigikogu (Estonian Parliament) election [133], making it the first national election where more participants voted online than on paper. The population has currently a good trust in this system, and this trust is translated by an important share of the population that used this way of voting for the last elections. Online voting is also preferred by citizens located abroad during the election (either living abroad or traveling). In 2019, 14000 votes were cast online from 143 countries, while in comparison, fewer than 1400 votes were cast from embassies and consulates [47].

Estonian voters authenticate using their national ID card. This ID card contains a chip with digital versions of the printed data, as well as two 2048-bit RSA keys and certificates associated with these keys. These keys are never communicated outside the chip, as the chip is capable of answering authentication challenges using the first key pair and generating digital signatures using the second key pair, using a card reader and a PIN code [13]. The certificates are valid as long as the national card is valid, and are revoked after the 5 years of validity of the card, or if the owner reports the card stolen or compromised [32].

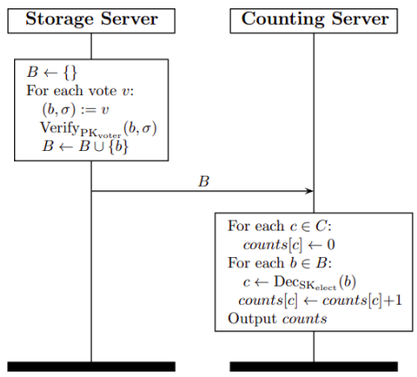
Voting framework until 2016: According to D. Clarke and T. Martens in *E-Voting in Estonia* [32] and S. Springall *et al.* in *Security analysis of the Estonian Internet voting system* [123], the voting process was as follows until 2016. Figure 4.1 represents the 3 phases of the voting process: vote casting (4.1a), vote verification (4.1b), and tally (4.1c) with the cryptography involved at each step. The client (on the voter's personal PC) connects to a Vote Forward Server (VFS) to initiate the process



(a) Vote casting process. [123]



(b) Vote verification process. [123]



(c) Vote tabulation process. [123]

Figure 4.1: Schema of the Estonia's I-voting system protocols (until 2016). From [123]

and authenticates using its ID card. The VFS sends the list of candidates based on the user's region. The user selects the candidates of her choice and the client generates an encrypted ballot with the election's public key and using a random number r. Then the user adds a signature using her ID card's second private key and its associate PIN code. The encrypted and signed ballot is sent to the VFS, which in turn forwards it to a Vote Storage Server (VSS). The VSS asks for confirmation of the signature to a Validity Confirmation Server (VCS), a tool that is used for all the official's digital services and is not limited to the election. The VCS compares the signature with the voter's certificate and informs the VSS whether or not the vote is valid. If it is the case, the VSS stores the encrypted ballot, and also writes the voter's identification number and a hash of the ballot into a log file. A receipt is then sent to the client, so that the user has the possibility to verify that her vote has been cast and registered as intended, by scanning a QR code using a verifying app. The QR code contains the encrypted vote, the ID of the voter, and the random number r used for the encryption. After retrieving the candidates list corresponding to the voter ID, the verifying app re-computes the encryption for each possible candidate and compares the outcome with the actual vote. When a match is found, the corresponding candidate is displayed to the voter. This possibility to verify the vote creates a risk of coercion (see Sections 4.1.3 and 4.1.4), which is mitigated by the fact that a voter can cast multiple votes, but only the last one is counted (all the previous ones are discarded). The online vote is also discarded in case the voter has also voted in a voting station. In order to avoid multiple times voting, the VSS compares the voter's identification number to the previous votes in the logs and revokes the previous encrypted ballots of the same user (this deletion event is also logged). This means that, at this moment, *the voter ID is linked to her encrypted ballot*. Later, the link between the voter and the ballot is deleted before the ballot is decrypted. However, the deletion is done in a procedural operation. When the voting period has ended, the VSS separates the digital signatures of all the ballots. The digital signatures are stored separately as proof of who has voted. The hash of each vote is

written in another log file, to keep track of all the votes which go for counting. The encrypted ballots without signatures are sorted by constituency, and then manually exported onto physical media hardware and transferred to the vote-counting application (VCA). The ballots are then decrypted on hardware security modules (HSMs) that contain the election private key. The HSM counts the votes and outputs the official results. The logs are later used for audit.

Critics of this model : This earlier Estonian I-voting system has received much criticism in the literature, on both theoretical and technical security concerns. According to [32], *The lack of traditional end-to-end verifiability causes voters to have to rely on secure hardware and carefully crafted security procedures to ensure the integrity of the vote.*

One of the most critical threats of the system is the association of the hash of the encrypted ballot with the voter's identification number. The models consider that anonymity is guaranteed thanks to the separation of the digital signatures with the content of the encrypted ballot, before the decryption phase. No one piece of hardware or user ever holds both the server's private key and a ballot with the digital signature attached. However, this separation is done by a procedure executed by the election's officials, and if a malicious administrator could make a copy of the encrypted ballot and their signature and later have access to the decryption's private key to recompute decryption, this would consequently lead the revelation of all voters' choice. This attack is prevented by legal and technical procedures, but the model is not safe by design.

In 2003 and 2010, the Estonian National Electoral Committee (NEC) produced experts report on security and risk analysis of the I-voting system [12]. The analysis raised fundamental problems and showed technical and architectural problems. Most of the issues where solved in the later versions of the i-voting model [123], but some fundamental concerns, such as the necessity to trust the voter's computer, cannot be totally avoided in the field of online voting.

Other security risks have been raised in [32], including the fact that the voter's computer communicates with both the ID card and the server (therefore a malicious program could request a voter's signature for another candidate than the chosen one and send this fraudulent but valid vote to the server - this could be nevertheless detected during the verification step) and that the logs on the server could be modified by an attacker (therefore if an attacker deletes some votes and their related logs, the attack would not be detectable).

In [123], Springall *et al.* analyzed the security of the Estonian Internet Voting System. They observed the 2013 municipal election with official accreditation, met election officials and developers, and reviewed the server's source code and written procedures. They identified a series of deficiencies related to poor procedural controls, lapses in operational security, and insufficient transparency measures. They observed that some procedures were not consistently followed, that procedures to handle anomalies were inadequate, and that the staff did not always know or respect such procedures. Many concrete examples are cited, for instance, due to a hardware problem, the election's results were transferred to an unsecured USB stick. The staff also used their personal laptop to realize sensitive operations. A lack of transparency was pointed out, and some minor vulnerabilities were found in the server's code.

Despite all this criticism, it is worth noting that, until now, no attack or failure of the Estonian i-voting system has ever affected an official election.

New voting framework (since 2017) : In response to these critics, the election administration system was restructured in 2017, separating oversight and management responsibilities between the National Electoral Committee and the State Electoral Office responsibilities. I-voting was integrated into Estonian's e-governance ecosystem, with an updated technological framework. According to [47], the new voting process is depicted in Figure 4.2 and works as follows : The voter first authenticates herself to the Vote Collector server (VC) from the official voting application and uses her eID-card

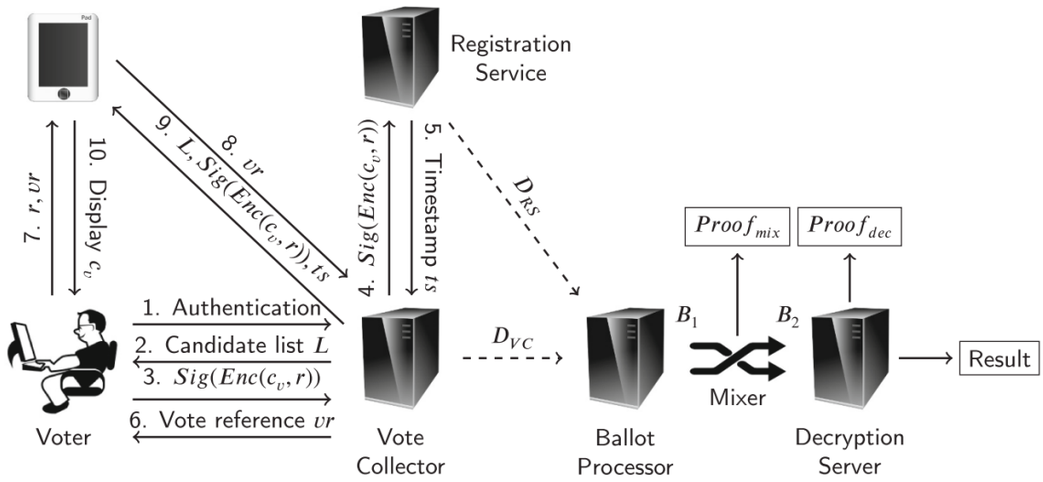


Figure 4.2: General scheme of Estonian voting process since 2017, from [47].

or mobileID. The servers respond with the list of candidates L corresponding to the voter's electoral district. The voter chooses her preferred candidate c_v, encrypts the vote with the Decryption Server's public key (and a random number r, so that votes for the same candidate look different), signs the cryptogram using her eID, and transmits the result to VC. To prevent any accidental or malicious omissions of votes by VC, the protocol requires the vote to be committed to a distinct Registration Service. This service then provides a timestamp ts as certification that the vote has indeed been securely transmitted outside VC. The voter can still verify that her vote has been cast and registered as intended using an independent verifying application. This app does not contain the decryption private key but instead recomputes the encryption for each candidate using r (randomizer) and a vote reference vr provided by the VC at the reception of the vote, and compares it to the vote until a match between the encryption of a candidate and the casted vote is found. To ensure an independent audit of the tally process while preserving vote anonymity, the resulting ballot list B_1 stored on the server is sent through a Mixer component that produces a mixed output ballot list B_2 along with a cryptographic proof of correct mixing. This new list is finally decrypted using the Decryption Server's private key, to produce the final result along with a cryptographic proof of correct decryption. Almost all the code of the i-voting system is open source and can be found in a Github[1] repository updated at each election. An exception remains for the official voting client app. It was justified as a defense mechanism considering the risk of malware on the voter's machines. However, the verification mobile app (which is open source) can be used to verify the correctness of the code's operation [47].

This new framework is relatively new with limited analysis available in existing literature. Our research did not reveal any significant criticism or opposition to this recent approach. However, these changes were praised by international observers such as the Office for Democratic Institutions and Human Rights (ODIHR), an institution part of the Organization for Security and Co-operation in Europe (OSCE) for improving system integrity and secrecy properties [105].

New South Wales's iVote system

The state of New South Wales (NSW) in Australia implemented the remote voting system *iVote* in 2011 with Scytl, a Spain-based company specializing in electronic voting and election technology. The system allows certain groups of voters (blind people, disabled, and people living far from a voting center or abroad) to vote online, complementing other voting methods like telephone, postal, and in-person voting. It has since been utilized in three state elections (2011, 2015, and 2019) and 17 partial

[1] `https://github.com/valimised/ivxv`

regional elections between 2011 and 2018 [132]. During the 2018 State General Election, 280.000 electors voted online [138].

Voters eligible for Internet voting must register beforehand and initialize a password. They authenticate online using documents proving their ID to the Document Verification Service (DVS) managed by the state of NSW. The encrypted votes are sent to a central server. The voter can verify her vote using a mobile app, by scanning a QR code displayed by the browser. The QR code contains the encrypted vote and the seed used for the encryption. The app can recompute the encryption using this information and the user's password. The system is therefore not receipt-free. To avoid vote coercion, the user can phone call the support of iVote to cancel her vote. She will then receive new credentials and may cast a new vote. When the voting period is over, external auditors verify that the ballots they received correspond to the ballot on the voting server, and verify the signatures and the validity of the Zero-Knowledge proofs (ZKPs). After their validation, the votes are sent through a mixnet to anonymize the votes.

The system is not Open Source. In 2019, Sctyl allowed experts to confidently analyze the code to detect any risk only 45 days before the election.

In [132], the authors consider that the votes can be considered confidential only if the client and server software are trustworthy, and if the person and machines involved in the decryption keys management can be trusted.

The NSW Electoral Commissioner has decided not to use iVote at the 2023 election due to a lack of time for testing and updating the software but claims that *«The decision not to use iVote at the State general election in 2023 has not been driven by any concerns about cyber security matters in previous elections»* [2].

I-Voting worldwide

As previously mentioned, Estonia is the country in the world where Internet voting is the most extensively used (more than half of the votes proceeded online in 2023's election [133]), and New South Wales also has its own system iVote (but it wasn't used for the last election in 2023). Other countries or regions around the world have also used i-voting or are developing pilots to do so, but usually at smaller scales.

For instance, between 2011 and 2014, Norway implemented a trial of remote voting, using a protocol designed by the commercial company Scytl, that included a combination of Internet voting, postal mail, and SMS. However, [44] considers that this system was not end-to-end verifiable. Norway stopped using this technology after the discovery of cryptographic failures, and because it did not have a significant impact on the turnout [132].

Different cantons in Switzerland also used an internet voting system between 2004 and 2008, developed by Scytl and the Swiss Post. The canton of Geneve abandoned this system in 2020 due to financial reasons [132].

In France, internet voting is available for citizens living abroad only. The voting system has a basis that is inspired by Helios but with a non-public bulletin board [35]. The voters use a Javascript client to authenticate with their personal data and a password, to form their encrypted ballot, and send it to the server. These ballots employ homomorphic ElGamal encryption and zero-knowledge proofs. The server assigns ballots to one of 708 precincts (a subdivision of a district, that corresponds to a physical polling station). At the end of the voting process, voters are invited to verify their vote. They receive a PDF file as a receipt that contains a hash of the ballot, the precinct ID, a signature of the server, and a link to the website where they can proceed to the verification using these data. The decryption key, split among 14 trustees, requires four for decryption. As the votes use homomorphic encryption, only the final results for each of the 708 precincts are decrypted, but not the individual ballots. The online voting phase concludes a few days before the physical polling day. Internet voters are then removed from the in-person voting list, making them ineligible to vote again at a polling station.

The papers *A review of E-voting: the past, present and future* [61] by Paul Gibson *et al.* and *E-Voting Meets Blockchain: A Survey* [138] by Maria-Victoria Vladucu *et al.* list many more countries to have ever used i-voting or to be planning to do so.

Washington, D.C. Internet Voting System testing failure

In 2010, Washington, D.C. initiated an Internet voting pilot project to enable overseas absentee voters to cast ballots through a website. Before implementing the system in the general election, the District conducted a distinctive public trial, inviting anyone to test or attempt to compromise its security. Within 48 hours of the system launch, a team of experts gained near-complete control of the election server, altering every vote and revealing almost all secret ballots [140]. Election officials remained unaware of the intrusion for nearly two business days, highlighting the practical challenges in securing online voting systems. After this trial, the Washington, D.C. Council decided to cancel the pilot and never implemented again another program of internet voting.

Helios

Helios [6] is an open-source and open-audit web-based voting system, first released in 2008. In March 2009 the Université catholique de Louvain (UCLouvain) used Helios to elect the university President [7]. Since then, the system has been used in numerous association elections such as the International Association for Cryptographic Research (IACR) board members election [70, 109]. The election admin submits a list of email addresses for each eligible voter. Each voter receives an email containing her confidential credentials and then proceeds to vote on the Helios web page. The vote is cast using public key encryption. A user can cast multiple votes, but only the last one is counted in the tally [44]. The user can verify that her vote is cast as expected by reproducing the cryptographic computations with the parameters given by the client. The user can also verify that the vote was registered as cast using the produced hash of the ciphertext [44]. Any observer can verify that all registered votes are tallied correctly. Therefore, it respects all three characteristics of end-to-end verifiability (see below). Helios uses homomorphic encryption so that the tallying computations can be performed on encrypted votes [109], with only the final result being decrypted. It also utilizes non-interactive zero-knowledge (NIZK) proofs to ensure votes are encoded and encrypted in a correct form and decrypted properly [10]. Helios depends on pre-instructed email addresses containing confidential credentials. Not only is this method insecure for national elections, but also it does not include any authentication feature, resulting in limited protection against insider ballot stuffing. A partial solution to this last issue would be to let a third party, such as the election authority, to proceed itself to the voter's authentication, and then send an email with the credentials to the provided email address. Also, homomorphic and NIZK may be computationally intensive, therefore limiting the scalability. If Helios is efficient in an election with up to a few thousand voters, it would be unable to support large-scale national elections [44].

The quest for End-to-End Verifiable Internet Voting (E2E-VIV)

In 2013, the US Overseas Vote Foundation, a nonprofit organization dedicated to overseas and military voter participation, launched the *End-to-End Verifiable Internet Voting: Specification and Feasibility Assessment Study* (E2E-VIV Project) [44], published in 2015. This study analyzed multiple Internet voting systems designed at that time and concluded that none of them was at the same time secure, usable, and transparent enough to hold public elections. They reclaim that any public election should be completely End-to-End Verifiable, and make recommendations in that sense (e.g. Recom. 2 : «*No Internet voting system of any kind should be used for public elections before end-to-end verifiable in-person voting systems have been widely deployed and experience has been gained from their use.*»). A. Essex [51] also argues for similar recommendations.

End-to-end Verifiability can be defined as follows [94, 44, 10]:

1. **Cast-as-intended**: A voter is able to verify that their intended candidate is correctly captured in the cast vote.
2. **Recorded-as-cast**: A voter is able to verify that their cast vote is correctly recorded by the system.
3. **Tallied-as-recorded**: A voter (or **any** observer) is able to verify that all recorded votes are tallied correctly.

According to the study, no i-voting system designed at that time was both E2E-Verifiable and respected the requirements of usability, security, and transparency (the authors considered Helios not usable because of its complexity of use, its risks of vote buying, and the need to register voter's email addresses). The study also makes recommendations on the cryptographic foundations of such a system, on the architecture of the protocol, and establishes a list of requirements to hold online elections. In *An Overview of End-to-End Verifiable Voting Systems* [10], Syed Taha Ali and Judy Murray provide an analysis of multiple E2E-Verifiable voting systems.

End-to-end verification, while beneficial, is not devoid of risks. As noted in [132], public ballot voting can lead to compromised voter privacy if there are errors in the encryption process, if the credentials or private keys are leaked, or if the cryptographic primitives are compromised.

4.2.6 Analysis of Belgian elections system

In this section, we will analyze how the requirements of an election are translated in the current Belgian voting system, and also in the model proposed by [132] to implement a system of mail-in voting using online elements for citizens living abroad.

Current voting system

Belgium's current voting process, previously outlined in Section 3.4 and further detailed in [45], incorporates both traditional paper voting and electronic voting with Voter Verified Paper Audit Trails (VVPAT).

In both modalities — paper and electronic — the *authentication* of voters is accomplished through presenting the convocation, distributed before the election by municipal authorities to all eligible voters. After casting a vote, a voter is marked off the list to avoid multiple voting instances (*one-human, one-vote*). Regarding *availability*, voting is conducted at designated stations during specified hours (opening at 8 am and closing at 2 pm for paper voting and 4 pm for electronic voting). Elections typically occur on Sundays to maximize voter turnout. Provisions for proxy voting exist for those unable to vote in person. Voters are assigned specific stations, which could pose challenges for those not in proximity, but proxy voting remains an option. The vote is of course *free* for every participant.

In paper voting, voter *anonymity* and *privacy* are safeguarded by the confidentiality of the voting booth. Given the uniform appearance of ballots, it's impossible to link a vote to a specific individual once cast in the ballot box. The *integrity* of the vote is the responsibility of election watchers, who ensure no unauthorized interference occurs with the ballots, and that all valid votes are accurately tallied.

In the case of voting on electronic machines, *privacy* is normally also protected as the vote also occurs in the voting booth, and the magnetic cards to activate the machines do not contain personal information on the voter. On *integrity*, as the names of the selected candidates are written on the paper printed by the machines, it appears to be the same as for paper voting. However, the papers are not always manually counted. Therefore, the machines implementing the electronic count must be well-designed. The trust is therefore also placed on the machine developers and the president of the station manipulating them. Thanks to VVPAT, voters can *verify* that their vote is indeed cast as intended. For the rest (tally as cast), the trust is based on the election watchers.

Analysis of the proposed model of mail-in voting for citizens abroad

In 2021, a study [132] charged to evaluate the possibility of implementing online voting in Belgium concluded that such a hypothetical system could not be securely deployed before 2034. However, the author proposed in the second part of the study a model of a mail-in voting system that includes online elements. This system could be used to improve the current system for mail-in voting for citizens living abroad (and could extend to people unable to come to a voting station, such as prisons, hospitals, and rest homes) and pose the basis for the development of online voting in the more distant future.

In the **current system**, complex logistics are necessary for printing and sending the voting ballots to all registered citizens living abroad. Upon reception of the ballot, the voter sends her vote using two envelopes. Here's how it works:

1. The voter places her ballot in an envelope. Along with these envelopes, the voter includes a form of identification. All these items are then gathered into another, larger envelope.
2. This larger envelope is sent to the president of the main office of the electoral district to which the voter belongs.
3. Upon receipt, the outer envelope is opened at the main office, and the identification form is examined. The voter's participation is then marked on the electoral roll.
4. Each envelope containing a vote is sent to the appropriate counting office.

This double-envelope system is necessary because the voter does not have any other means to prove her voting rights except by attaching an identification form to her ballot. This process ensures the integrity and confidentiality of the vote, allowing for identity verification while maintaining ballot secrecy. However, confidentiality depends on the main office which must transmit the inner envelope without opening it. If the vote is intercepted, the choice of the voter will be revealed.

In the **proposed model** of a mail-in voting system with online elements, the voter first authenticates using an electronic method, such as her eID or the mobile app itsme (see Section 3.5). She then receives a blank ballot in PDF format along with a list of unique codes. She prints the blank ballot, fills it with the candidate of her choice, and signs it. The codes received after authentication are not mere signatures but are part of a cryptographic system designed to ensure the integrity and authenticity of the vote. Each code is uniquely assigned to each voter and indicates either the selection or the non-selection of a candidate on the ballot. It links the filled ballot cryptographically to the authenticated voter without revealing the voter's identity.

The blank ballot contains a random 128-bit token displayed on each page as a string or as a QR code. This token plays a dual role. Firstly, it authenticates the eligibility of the voter by linking the ballot to a token generated by the server, which, crucially, does not retain a record of the voter associated with each token. Secondly, it allows the voter to verify later that her vote was registered correctly and was not lost by the postal service.

The fact that the voter could print the ballot herself simplifies the logistics, as there's no need for the blank ballot to be physically sent to her. Once the office receives the vote, it is opened, encoded electronically, and then sent securely to the election authority. Moreover, the entire operation requires mailing only from the voter to their consulate, significantly reducing the logistical burden. The aim of printing a blank ballot to be filled in by hand, and not already pre-filled, means that it can be printed even in a conventional print shop, or on a shared printer, without fear for its confidentiality. The unique codes and the cryptographic token ensure the confidentiality and integrity of each vote, even when ballots are printed in less secure environments.

4.2.7 Conclusion of the comparison

We have highlighted in the previous sections that no voting system is perfect, each having its own drawbacks. Despite being widely used in many countries for a long time,

the traditional voting system using paper does not totally prevent vote manipulation by a corrupted government, requires complex logistics, and is organized at a huge cost. Such a cost might seem reasonable to preserve democracy. But it prevents from organizing regular votes such as early elections to solve a political crisis, or referendum and popular consultations, which are central pieces in participating democracies.

Electronic voting, on-site, does not solve the problem of the cost, making it even worse with the buying and maintenance of expensive machines. It might prevent fraud on the condition of being transparent with the code, but the verification can solely be done by experts. The security and fairness of such systems cannot be guaranteed by design and rely on the expertise of a few experts who will audit the process before and after the election. After multiple elections using DRE for electronic voting in different countries, the population often had low confidence in such a system, and some countries turned back to traditional voting. The use of a voter-verified printed audit trail (VVPAT) like a human-readable indication of the vote printed on a paper offers a guarantee that her vote has been cast as intended and therefore increases the confidence of the population regarding e-voting. However, it does not offer a guarantee that the vote has been recorded as intended. This property still relies on the trust regarding the tallying machines, and the humans working at the voting station. Note that a proof of vote recorded as intended would violate the requirement of receipt-freeness and would raise risk of vote coercion, as discussed in Section 4.1.3.

Remote voting partially solves the problem of the cost of organizing an election, but the cost to initially develop such a system must be taken into account. It also increases accessibility for the electors who may encounter difficulties for any reason (leaving abroad, living far from a voting station, illness, not being able to move outside) to come to vote into a voting station. However, remote voting, and especially online voting, brings many new risks, including cybersecurity risks related to the client device, to the server, and to the network. Large attacks, potentially undetectable are henceforth possible. In case of infection, the central server is at risk of getting its registered votes, if not modified, simply deleted. Also, a DoS attack would lead to unavailability. Finally, online voting requires trust in the election authority. A corrupted government could attempt to discretely modify the votes on the server or add additional fraudulent votes. A partial solution to these problems is universal end-to-end verifiable voting systems. End-to-end verifiability means that a voter can verify that her vote is correctly encoded and registered on the server, and any observer can verify that all registered votes are correctly tallied.

McCorry *et al.* [94] suggest an initial step of replicating voting data on mirrored websites in diverse locations to counteract the potential of a single compromised server altering data. This approach makes undetected data manipulation considerably more difficult, as changes would need to be replicated across all these sites. However, they propose blockchain as a more effective solution, as it synchronizes replicated ledgers across a network of peers, ensuring consensus and enhancing data integrity.

Blockchain could indeed be a solution to solve some of the aforementioned problems. By providing a decentralized and tamper-resistant ledger, blockchain can ensure the integrity of voting records, making it significantly harder for malicious actors to manipulate results. Additionally, the use of smart contracts on the blockchain can automate certain aspects of the voting process, further reducing the risk of human error or interference. As the votes are publicly readable, the results are publicly verifiable, enhancing transparency. However, it does not prevent all the risks related to cyber security (DOS, phishing, malware on the client side, ...) and even brings new risks. The analysis of the use of blockchain for online voting is done in the following section.

4.3 Analysis of blockchain-based voting system

A detailed description of blockchain is provided in Section 3.2. Simply put, it is an append-only data structure where transactions are stored together in blocks. Each block contains the hash of the previous block, therefore forming a chain. The chain is only valid if the hash of each block corresponds indeed to the value contained in

the next block. This property guarantees the integrity of the data. The blockchain is distributed across multiple nodes in a P2P network, making it into a distributed ledger. When the blockchain is openly accessible, it ensures transparency, enabling anyone to read the ledger data.

In a blockchain-based online voting system, voters can vote remotely and cast their votes on the blockchain. The main benefits of this method are transparency (as anyone could verify the correct tally of the votes), integrity (as the data cannot be altered), and security (as the database is distributed). However, this transparency implies new challenges to guarantee the anonymity of the voters and the respect of all requirements.

4.3.1 Adaptation of voting requirements

Let's analyze how each requirement of an election adapts to blockchain-based voting.

Generic requirements :

- Integrity: Integrity in the blockchain is ensured through the data structure it employs. Any attempt to alter the data within a block would result in a change in the block's hash value. Since each subsequent block contains a reference to the previous block's hash, any tampering in a single block would render it invalid, cascading this invalidation through subsequent blocks in the chain (see Section 3.2.1 for the blockchain data structure definition).
- Privacy/Anonymity: It is a big challenge, as the blockchain's data is publicly readable. The use of cryptography is required to guarantee the property of anonymity.
- Authenticity/Eligibility: The system must only accept votes from eligible voters.
- One human, one vote (un-reusability): The system must ensure that eligible voters can only vote once.
- Availability/Timing: It is straightforward to refuse any votes cast after a predefined time. However, a difficulty remains in the unveiling of the results, which should be kept secret until the end of the voting period.
- Auditability: As all the transaction are logs, it is easily auditable.
- Freeness: the vote must be free. Most of the public blockchains depend on transaction fees in their economic model. This cannot be the case, or at least, voters shouldn't pay by themselves these fees. For people who do not possess a device to vote online, there must be a possibility to vote on-site.
- Security: Additionally to threats from the traditional vote, and especially to threats from e-voting (malware on clients or nodes), and remote voting (DoS, etc), blockchain-based voting may be victim of new types of attacks. The model has to be protected from these threats (see later).

e-voting specific requirements

- Accessibility: As for i-voting, non-tech-savvy people should be able to cast a vote easily, and it should be compatible with most of the connected devices. The use of blockchain technology shouldn't make the voting process more complicated.
- Open source: Most blockchains or blockchain frameworks are open source. The Smart Contracts (see Section 3.2.5) should also be open source.
- Logs: The blockchain acts as a ledger where all transactions (= the logs) are stored.
- Transparency: One of the reasons why using a public blockchain, is because it offers transparency, as anyone can verify the current state of the blockchain.

i-voting specific requirement

- Receipt-freeness: the system design must ensure that the user cannot get a receipt of its vote. In other words, the user cannot redo the computation to get the content of its vote.
- Coercion resistance: This property faces the same challenges as for remote voting in general.
- Hybrid internet-paper vote: as for i-voting, it is better to have both systems that can be running in parallel.

4.3.2 Capacity, Security, and Decentralization requirements

The Blockchain Trilemma (see section 3.2.9) prevents a blockchain from being at the same time totally decentralized, secured, and scalable. We will now analyze how to balance between these three properties to find a convenient equilibrium.

Capacity requirement

The criteria of scalability is very important when we need to have a lot of interaction with the blockchain. Scalability in blockchain refers to the capacity of a blockchain network to handle a growing amount of work or transactions. It's a crucial feature that determines how efficiently a blockchain can grow and serve more users or handle more transactions without compromising performance, speed, or cost. In distributed databases, handling a large number of modification of the ledger is a challenging task, as this ledger must be shared among all the nodes of the network. We will now try to estimate the capacity requirement of our system (i.e. the number of transactions per second that are processed by the blockchain). During the 2018's election, 8.167.709 people had to vote in Belgium. The polling stations are open from 8 a.m to 4 p.m, so a total of 8 hours of opening time. This gave an average of 1.020.000 people per hour. If we consider that the voter flow is not uniform and there are peak hours, we can imagine a peak of 2 million voters in one hour. This gives an average of 555 votes per second. To keep a large margin, we can target that our model needs to handle at least 1000 votes per second.

Security requirements

Election is an extremely high stakes, so there can't be any compromise on security. An analysis of the risks of blockchain-based voting is done at the end of this section.

Decentralization requirement

The last property of the Blockchain Trilemma (see Section 3.2.9) is decentralization. Decentralization refers to the distribution of computing nodes across a network in a way that minimizes central control or authority. This distribution of nodes helps enhance resilience, security, and the ability of the network or system to continue functioning even if some nodes fail or become compromised. The difficulty of such a distributed data structure is that all the memory must be shared and synchronized, at all times, between all the nodes of the network. Some decentralization is needed to avoid a single point of failure or to prevent a dishonest node from taking control of the blockchain. It also reduces the connection load on each server. In a fully decentralized network, there is no central authority. In that case, the entire network is managed by all the stakeholders together. However, in the case of an election, the absence of a central authority is not a targeted property. Indeed, at the end of the day, this is the election official who organizes and manages the election. A complete lack of central authority would open the door for any entity to assume control over the election, which is undesirable.

So we need a central authority, that acts transparently and anyone can verify its fairness. A question that remains is who can manage a node (public institutions, private companies, universities, ... ?) and on the geo-localization of such nodes. If all the nodes are localized in the election's countries, it is good for national sovereignty but is at the risk of attack against the national telecommunications networks (same for remote voting).

In summary, addressing the Trilemma necessitates exceptionally robust security, substantial scalability, but tolerates a reduced degree of decentralization.

4.3.3 Risk with blockchain-based voting

The use of blockchain for an online voting system was supported by the idea that it could be more resilient than a centralized server, and offer more transparency than a "black box" operated by the government. However, blockchain technology cannot

solve all the problems related to online voting, and might even bring new ones. In this section, we present a non-exhaustive list of the security risks related to blockchain voting.

Risks inherent to online voting

First of all, among the problems brought by online voting that cannot be solved by blockchain, we can site the risks of malwares infecting the client machine, and its risks of alteration, unauthorized disclosure, or non-transmission of the user's vote. Also, phishing attempts could redirect the voter to fake websites imitating the appearance of the official election website, to steal her credentials, preventing her from actually voting or revealing her voting intentions. On the other hand, decentralization offered by blockchain, even with a moderate level of distribution, can help enhance the robustness of the system, compared to traditional online voting. Distributed Denial of Service (DDoS) attacks become more challenging due to the decentralized structure, requiring significantly more resources to disrupt the network. Unless an attacker could take control of the network, it couldn't alter or delete any votes. At most, a DDoS could try to block the issuance of new votes, but it is much more complicated to achieve than compared to a centralized network. Note that the same improvement can be obtained with a distributed network even without using blockchain. Blockchain does not just bring availability, but also integrity and constituency. A DDoS attack targeting individual voters is unlikely to be effective on a large scale, as such an approach would be resource-intensive and inefficient. In contrast, a more substantial threat is a DDoS attack on national internet infrastructure. This type of attack could disrupt internet voting by affecting broader connectivity, impacting the ability of voters to access the voting platform. This vulnerability is inherent to any form of internet-based voting and is not specifically addressed by the use of blockchain technology. Blockchain enhances network robustness but cannot mitigate risks associated with the broader internet infrastructure's reliability and security.

Additional risks brought by blockchain

The use of blockchain brings its own new risks. The most famous type of attack on a blockchain is probably a *51% attack*. This is an attack that concerns permissionless blockchain, usually using the Proof-of-Work (PoW) or Proof-of-Stake (PoS) as consensus protocol. In this type of blockchain, like Bitcoin [100], any entity can join and participate in the network, no matter the localization. A 51% attack occurs when a single entity or group gains control of more than half of a blockchain network's mining power, computing power, or stake, depending on the consensus mechanism used. In this situation, the attackers, having majority control, can influence the network's consensus decisions. They can potentially reverse transactions, leading to double-spending, or prevent all new or targeted transactions from being added to the chain. This type of attack would require astronomically high computational power to take control of large, well-established blockchains, but smaller or less-secured blockchains could be more vulnerable to this type of attack.

Sybil attacks (not limited to blockchain but in peer-to-peer networks in general) occur when a single adversary controls a network or a protocol by creating multiple fake identities. In this attack, what appears to be a multitude of unique users is actually multiple identities controlled by a single entity. This can happen in a voting protocol where each user gets one vote, but an attacker has an opportunity to create multiple identities.

Some *wallet-based* attacks target individual users to steal their funds, like malware disclosing private keys, or scams making a user sign a fraudulent transaction that transfers assets to the scammer's wallet. Attacks *on smart contracts* often exploit code vulnerabilities, such as reentrancy bugs or integer overflows, to lead to an undesirable effect, such as stealing assets in the domain of Decentralized Finance (DeFi) for instance. Note that some of these risks concern especially financial transactions (e.g. double-spending) but would not make sense in the context of voting, while some are

more general like Sybil attacks or exploitation of vulnerabilities in Smart contracts. T. Guggenberger *et al.* realized a structured overview of attacks on blockchain systems, reviewing 161 papers released between 2013 and 2019 analyzing attacks targeting the entire blockchain technology stack, from P2P network to application logic.

Risks of blockchain based-based voting systems in the literature

The use of blockchain for voting systems is a trending topic in the literature with polarized opinions. For many, like by Hyunyeon Kim *et al.* in [83], blockchain improves the online voting system by adding transparency, integrity, and security. This is also the case for B. Shahzad and J. Crowcroft in [120], and for many more, including the papers analyzed in the next section. On the other side, some papers argue that blockchain should not be used for voting. For instance, in the article with the encouraging title *From bad to worse: Internet voting to blockchain voting* by Sunoo Park *et al.*, the authors consider that using blockchain for voting does not solve the major problems of i-voting, but brings new ones. For instance, they criticize decentralization, arguing that elections are inherently centralized by the election authority. According to them, decentralization comes with potential congestion and difficulty in upgrading. Because of difficulties in governance and coordination, these systems are supposedly slow to react when facing vulnerabilities. This paper raises important points but has some limitations. It only considers some basic configuration of the technology and ignores the possible patches easy to implement (e.g. the paper states that «*permissioned blockchain is inaccessible for users, therefore the users can't verify that their votes were registered* », while it suffices to permit some nodes to share publicly the data). Also, the paper declares somewhere that «*online voting led to slight decreases in turnout after its implementation in Belgium* ». However, Belgium has never experimented online voting. In reality, the original source [38] said that turnout in Belgium has slightly decreased since the introduction... of electronic voting machines, which is a completely different fact. In [85], the authors argue that the same level of security and transparency could be achieved without using a blockchain.

Chapter 5

Literature review

The previous chapter formalized the requirements of an election (in particular online elections), highlighted the risks of such online voting systems, and introduced the concept of a blockchain-based online voting system. This chapter aims to find an ideal model of a blockchain-based voting system that could fit the requirements aforementioned. First, a naive example with a simple Smart Contract is presented (Section 5.1). This example will outline a general solution with the associated voting process. While easy to implement and understand, this example has multiple problems, especially the anonymity of the voters which is not well respected. Therefore, an exploration of the solutions offered in the literature is done in Section 5.2, analyzing 6 distinct solutions. At the end of the chapter (Section 5.3), a comparison of these models is conducted.

5.1 Naive example with Smart Contract

Before exploring the numerous complex models proposed in the literature, it's instructive to examine a fundamental yet functional solution that utilizes a Smart Contract. The code for this Ballot Smart Contract is provided in Appendix A.1. For better visualization, some methods of the code are displayed in Figure 5.1.

Smart Contracts are defined in Section 3.2.5. Simply said, these are automated programs that run on a blockchain, executing predefined actions. This open-source contract, written in Solidity (an object-oriented programming language used for implementing smart contracts on Ethereum Virtual Machine (EVM) compatible blockchains (see Section 3.2.4)), implements a standard voting system on the EVM. While originally designed to accommodate vote delegation, this adaptation simplifies the model, assigning a weight of 1 to all eligible voters, including the chairperson.

The `voters` mapping associates public addresses with the `Voter` struct, and the `proposals` array contains the list of candidates. The `giveRightToVote` function grants a weight of one to eligible voters. This function can only be called from the address of `chairperson`, which corresponds to the address that originally deployed the contract. The `vote` function ensures that a voter has the right to vote and has not voted previously. It then records the vote, incrementing the chosen candidate's score by 1. The `winningProposal` function determines the proposal with the highest vote count, while the `winnerName` function returns the name of the winning proposal (not displayed on Figure 5.1, refer to the complete code in Appendix A.1).

In the hypothetical case where this method is used for an election, the process would be as follows :

1. The election authority creates and deploys the Smart Contrat on the blockchain. Its public address is therefore the `chairperson` of the contract.
2. A voter creates a wallet, generating a pair of private and public keys, and its associate public address.
3. The voter authenticates to the election authority. In return, the authority calls the function `giveRightToVote` to give the voting right to the public address of the authenticated voter.
4. The voter calls the function `vote` to vote for the candidate of her choice.

```solidity
/* @dev Implements voting process*/
contract Ballot {

    struct Voter {
        uint weight; // weight could be accumulated by delegation
        bool voted;  // if true, that person already voted
    }

    struct Proposal {
        bytes32 name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    mapping(address => Voter) public voters;

    Proposal[] public proposals;
```

(a) Contract `Ballot`

```solidity
/**
 * @dev Create a new ballot to choose one of 'proposalNames'.
 * @param proposalNames names of proposals
 */
constructor(bytes32[] memory proposalNames) {   infinite gas 524800 gas
    chairperson = msg.sender;
    voters[chairperson].weight = 1;

    for (uint i = 0; i < proposalNames.length; i++) {
        // Creates a temporary Proposal object and
        //appends it to the end of 'proposals'.
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}
```

(b) `constructor` method

```solidity
/**
 * @dev Give 'voter' the right to vote on this ballot.
 May only be called by 'chairperson'.
 * @param voter address of voter
 */
function giveRightToVote(address voter) public {   29325 gas
    require(
        msg.sender == chairperson,
        "Only chairperson can give right to vote."
    );
    require(
        !voters[voter].voted,
        "The voter already voted."
    );
    require(voters[voter].weight == 0);
    voters[voter].weight = 1;
}
```

(c) `giveRightToVote()` method

```solidity
/**
 * @dev Give your vote to proposal 'proposals[proposal].name'.
 * @param proposal index of proposal in the proposals array
 */
function vote(uint proposal) public {   infinite gas
    Voter storage sender = voters[msg.sender];
    require(sender.weight != 0, "Has no right to vote");
    require(!sender.voted, "Already voted.");
    sender.voted = true;

    proposals[proposal].voteCount += sender.weight;
}
```

(d) `vote()` method

Figure 5.1: Selected methods of a basic ballot Smart Contract in Solidity. The complete code can be found in Appendix A.1.

5. Anyone can call the function `winnerName` to get the result of the election.
6. Any node operator with a copy of the blockchain can verify the validity of the chain and the result of the contract.

Despite its simplicity, this contract works very well to proceed to a vote on the blockchain. For instance, it might be used when a DAO (Decentralized Autonomous Organization, see Section 3.2.8) organizes a vote among its members. Thanks to its inherent properties, the blockchain infrastructure guarantees the security, integrity, availability, and transparency of the vote. However, this basic method is not suitable for an official election (otherwise, this master thesis would have been only two pages long). Indeed, the requirement of privacy/anonymity is not respected. The blockchain, publicly readable, can be seen as a distributed ledger of the logs of the contract, where every transaction (*i.e.* a call to a function that modifies the state of the contract, such as the `constructor`, `giveRightToVote`, `vote`) is stored in clear. For instance, a transaction calling the `vote` function contains particularly the public address of the sender (*i.e.* the user who initiates the transaction), and the parameter `proposal` (the chosen candidate). The address of the voter is therefore associated with the content of her vote. As long as the real identity of the voter is kept secret, the privacy of the voter is respected. The term to describe this concept is *pseudonymization*. However, it is impossible to guarantee this confidentiality. How could the voter be totally certain that nobody, including the authority staff or the government itself, can establish any link between its real identity and its address? Even if strong procedures are created to separate the different steps of the authentication and the provision of the voting right, the system would still suffer from a - justified - lack of trust from the population.

This requirement of anonymity is the hardest property to achieve when all the data is public. Consider a scenario where voting occurs openly in a town square. Voters

pick their candidate's name for all to see and place it in a transparent ballot box. This approach ensures absolute transparency, yet leaves no room for voter privacy. But, what if we could maintain both transparency and privacy? Consider now a voting booth where voters privately choose their preferred candidate, place their choice in an envelope, and then publicly deposit this envelope into a transparent ballot box. Here, the vote remains confidential, yet the process is transparent to all, ensuring no fraudulent votes are added. This is what happens in the traditional model, except that the public square is replaced by a voting station under the watch of election attendees, for practical reasons. Well, in the context of blockchain voting, the role of the envelope is achieved by cryptography. The vote is first encrypted so that the content of the vote cannot be associated with the voter and then transmitted publicly on the blockchain.

5.2 Literature review: methodology

In the following sections of this chapter, we will delve into various blockchain-based voting system models proposed in the literature. Our objective is to identify a model that aligns closely with the requirements outlined in the preceding chapter, balancing critical aspects such as transparency, anonymity, verifiability, and others.

We begin by surveying existing literature reviews, including:

1. *Systematic Review of Challenges and Opportunities of Blockchain for E-Voting* [129] conducted by Ruhi Taş and Ömer Özgür Tanrıöver.
2. *E-Voting Meets Blockchain: A Survey* [138] conducted by Maria-Victoria Vladucu, Ziqian Dong, Jorge Medina, and Roberto Rojas-Cessa.
3. *Blockchain for Electronic Voting System—Review and Open Research Challenges* [78] conducted by Uzma Jafar, Mohd Juzaiddin Ab Aziz, and Zarina Shukur
4. *A Review on Distributed Blockchain Technology for E-voting Systems* [117] conducted by Rihab H. Sahib and Eman S. Al-Shamery
5. *Blockchain voting: A systematic literature review* [40] conducted by Marianne Dengo
6. *Analysis of Blockchain Solutions for E-Voting: A Systematic Literature Review* [19] conducted by Ali Benabdallah, Antoine Audras, Louis Coudert, Nour El Madhoun, and Mohamad Badra

Each review adopts a different method for comparison. However, to filter out models, any model identified by authors of the reviews as having significant shortcomings in essential areas like integrity, anonymity, verifiability, or scalability is excluded. It's important to note that these reviews do not uniformly evaluate all these requirements, meaning some models that do not fully meet our needs may initially pass through our filter. Additionally, we conducted our research to discover papers not included in these reviews.

Many papers analyze the feasibility, the risks, or other aspects of voting systems based on blockchain, but do not propose a specific model. From the papers analyzed, we exclude those not focused on blockchain-based online voting systems for elections. For example, some papers discuss blockchain voting within DAO ecosystems or for less critical votes (*e.g.* associations, and student clubs), which do not align with our research objectives. Similarly, we set aside papers that utilize blockchain only in certain aspects of the electoral process but do not wholly base their system on blockchain or pursue aims analogous to ours. An example is the model proposed in *Blockchain-Based Anonymous Voting System Using zkSNARKs* [99] by M. Murtaza *et al.*, which, though it offers integrity, anonymity, and public verifiability, requires on-site voting rather than online.

After applying this first filter, we have identified multiple models that offer blockchain-based online voting systems that appear to be specifically designed for elections. We selected 6 of these models which proposed unique cryptographic protocols or architecture. In the following sections, we will conduct a detailed analysis of each of these

models.

For each, we start by summarizing their paper, describing shortly how they work, especially from a cryptographic and voting process perspective. Then, we analyze whether each requirement defined in Section 4 is respected or not, particularly *integrity, anonymity, authenticity, eligibility, unreusability* (one-human, one-vote), *availability, coercion-resistance,* and *verifiability.* We also point out any issue we see and indicate any information we consider relevant. When possible, we also evaluate the *scalability.* Given the rapid evolution of the blockchain voting domain of research, we will note the year of publication for each paper referenced in this literature review. Figure 5.5 synthesizes the results of the analysis.

5.2.1 Hjálmarsson et al. : Smart contract on permissioned blockchain

Hjálmarsson *et al.* proposed in *Blockchain-Based E-Voting System*[76], published in 2018, an e-voting system based on Smart Contracts, running on a permissioned blockchain. With over 500 citations according to Google Scholar, this model is a reference in the field. In their model, the election process works as follows :

1. **Election setup:** Authorized institutions run a node on a permissioned blockchain, that uses a consensus protocol (see Section 3.2.3). The paper does not discuss which institutions should be allowed to run a node nor the consensus protocol to use but simply says that these institutions should be «trusted ». An election administrator creates Smart Contracts for each election's district.
2. **Voter registration:** Eligible voters must register prior to the election. The country's official administration authenticates and authorizes eligible electors by providing credentials that allow them to vote in the district associated with the voter. A unique wallet is generated for each voter for each election that the voter is eligible to participate in.
3. **Casting vote:** To proceed to vote, eligible voters will simply interact with the Smart Contract using their wallet. They create a transaction that contains the transaction ID (hash of the transaction), the block number where the transaction is located, the Smart Contract address (therefore indicating the district ID), and a value corresponding to the content of the vote.
4. **Tallying results:** The Smart Contracts compute their tally on the fly for each new vote (by simply incrementing the number of votes for the selected candidate).
5. **Verifying votes:** Using the transaction ID, a user can verify her vote on an official election site (after authenticating themselves electronically) to see that her votes were correctly listed and counted.

The paper also considers 3 distinct blockchain frameworks that already exist and are usable out-of-the-box (Exonum, Quorum, and Go-Ethereum (Geth)), and compares their consensus protocol, their transaction throughput, their flexibility on decentralization and programming, and other properties. Their final choice is to use Go-Ethereum, an Ethereum implementation developed for private and permissioned blockchains, compatible with EVM (see Section 3.2.4).

Analysis of the model : This model is an easy-to-understand and easy-to-implement model, that indeed implements a voting system for elections. It is in many aspects very similar to the naive contract presented above. Additional parts such as the elector authentication and verification by the election officials are not detailed in the paper.

The *integrity* of the ballot is guaranteed by the tamper-proof property of the blockchain. *Authenticity* and *eligibility* are under the responsibility of the official authority that provides those credentials. The *one-human, one-vote* property is respected, as a wallet only allows to proceed to one vote, and the fact that a voter receives only one wallet is, as for eligibility, under the responsibility of the election authority.

However, one major issue with this model concerns *privacy/anonymity.* Standard blockchain transactions are linked to a public address and authenticated by a private

key, allowing for the verification of the signer's identity. In blockchain-based voting, like in the naive smart contract presented at the beginning of the chapter, this could expose voter identities as their addresses would be visible with their votes. The model here proposes not storing voter addresses in transactions, but the implementation details are unclear. If the nodes running the network receive the public address and remove it, then they could have the ability to illegally store the public address associated with the vote, therefore not guaranteeing anonymity. Later, it would be impossible to verify that the transactions accepted by the network after the consensus algorithm are indeed sent by legitimate voters. Another issue is that the *tally* is done on the fly (*i.e.* the results are known immediately, not at the end of the election, as is required in most elections). The model relies solely on the cryptography provided by the blockchain infrastructure, but the votes are not further encrypted, creating a potential link between addresses and the content of the vote, which are publicly readable in clear. Individual verification in the model requires authentication, presenting an added risk for voters in potentially leaving a trace of their identity, either through the authentication method used or via their IP address, which might be linked to the content of their vote. The possibility for voters to verify their vote (a form of 'receipt') combined with the inability to alter the vote post-submission introduces a *risk of coercion*. Furthermore, the model does not allow public *verifiability*, thereby falling short in terms of transparency. The paper does not provide information on the security aspect of the model (except the statement that blockchain offers integrity). It is subject to the same threats as other online or blockchain models. Compared to public blockchains, the authors argue that permissioned blockchain protects against 51% attacks, avoids high transaction fees (which allows the freeness of voting), and is not dependent on the variability of the throughput in case of important traffic. Permissioned blockchains, in this case, Geth, usually offer greater *scalability* than their permissionless equivalent, with high transaction throughput.

5.2.2 Kirillov et al. : Hyperledger Fabric Permissioned Blockchain using Smart Contract and Identity Mixer

Kirillov *et al.* proposed a model based on permissionned blockchain architecture with smart contract, using the Hyperledger Fabric framework in *Implementation of an E-Voting Scheme Using Hyperledger Fabric Permissioned Blockchain* [86], published in 2019. This model is based on O. He and Z. Su's e-voting scheme with blind signature detailed in [73], which adheres to requirements of eligibility, unreusability, untraceability, unduplicatability, unchangeability, and verifiability. Kirillov's model extends this concept by integrating it with distributed ledger technology to automate the electoral process.

The reader might want to first be familiar with the architecture of Hyperledger Fabric's network that is described in Section 3.3. The network is configured by the admin of the central voting authority (called Org) who determines the rights (read, write), the number of nodes (peers) that every organization has, and the ordering nodes that will include transactions into the ledger after reaching an agreement based on the consensus algorithm. Then it deploys the chaincode (smart contract) CC on the ledger. The chaincode plays the role of both the authority and the tallier in the He and Su's scheme [73]. The election authorities can be divided into departments. Each department Dep is a separate organization (in terms of Hyperledger Fabric) approved by Org and running at least one physical node. Each Dep generates a pair of public/private keys. The public key is stored on the blockchain, and the private key is saved in the private data collection (a special mechanism of hyperledger, the data is only accessible by authorized nodes).

The users must register prior to the election to be able to vote online. The unregistered users can naturally still vote on-site. A registered user can cancel her registration to vote onsite. The registration phase is done in two steps and is depicted in Figure 5.2.

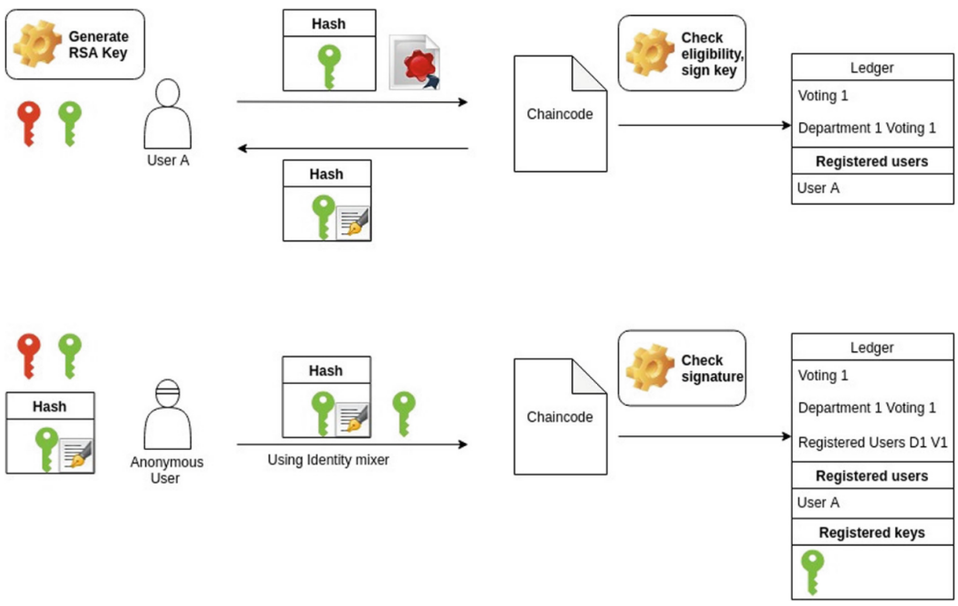


Figure 5.2: The user registration process. From [86]

In the first step, the voter requests a blind signature (see Section 3.1.3). She generates a key pair (E_v as public and D_v as private, but keeps both secret for now) and a secret random number R. She calculates $E_{dep,i}(R) * h(E_v)$, where $E_{dep,i}(R)$ is R encrypted with the department's public key, and $h(E_v)$ is the hash of the voter's public key. This result is sent to the chaincode (CC). If the user is eligible, the CC signs the data with the department's private key ($D_{dep,i}(E_{dep,i}(R) * h(E_v))$), sends it back to the user, and records on the ledger that the user has received a blind signature. The user multiplies it by the inverse of R to keep only what will be used as the signature of the public key by the CC.

$$D_{dep,i}(E_{dep,i}(R) * h(E_v)) = R * D_{dep,i}(h(E_v))$$

In the second step, the user sends anonymously the signature $D_{dep,i}(h(E_v))$ to the CC, along with their public key E_v, using an identity mixer (idemix). The CC checks the hash and signature and, if correct, records the public key E_v in the ledger. The ledger now stores the registered voter's public key and the fact that the voter has received a blind signature, but cannot associate both. This ensures voter eligibility without revealing their identity. The paper additionally explains a method for revoking the registration of a public key. However, this process falls beyond the scope of this summarized overview of the model.

During the voting phase, voters who have registered their public key can cast their vote. The vote is encrypted using the voter's private key and signed using its public key, sent through a mixnet, and then the CC (who receives an encrypted anonymous vote) verifies the signature and checks that the public key was registered. If this is the case, the vote is stored on the ledger. After voting, users are required to submit their private keys for decrypting ballots and tallying the votes. Once the keys have been uploaded, the CC decrypts all the votes and computes the results of the election.

Analysis of the model A major issue with this model is that in order to proceed to vote, the voter must first register, then cast a vote, and finally send the private key, at 3 distinct times. If it stops before finishing the process, the vote can't be decrypted and is therefore considered invalid. During the election, inspectors (or auditors) can run their own node with read permission to store a copy of the blockchain locally in order to detect cases of cheating.

Regarding *eligibility*, the model assumes that every eligible voter received a certificate (e.g. X.509) from the election authority. *Unreusability* (one-human, one-vote) is guaranteed by the fact that to register a public key, voters need a signature from the CC of their Dep, but only one signature is sent per eligible voter. The use of blind

signature and Identity Mixers assures the *anonymity* of the voters, as there is no link between a vote and the identity of the voter. The voter can *verify* that her vote was counted. However, the model does not respect *receipt-freeness*, as the voter can prove that a certain public key belongs to her. The authors suggest employing ring signatures to address this limitation in future work.

The original paper does not delve into discussions regarding *scalability*, showing no implementation in practice, and providing no empirical data to assess its performance in real-world scenarios. However, the authors published a performance analysis of their model in 2020 in [85]. They express doubts regarding the use of blockchain for their model. They run an experiment of their model, using two network configurations: one with only one organization, running one orderer and one peer (so it is a completely centralized model), and one with 2 organizations, running each 2 nodes and with 2 orderers in total, making it still pretty much centralized, but the update of the ledger requires the computation of a consensus algorithm. The results of the experiments show that both networks cannot handle more than 2000 tps, and a maximum of 1300 simultaneous active users. The authors claims that they can achieve the same level of confidentiality using a centralized central more efficiently, and conclude by saying that is is *«advisable to abandon the use of the blockchain without losing the security properties »*. They add that even with distributed ledger technology, the election authority retains the responsibility for issuing certificates. While the system effectively checks each voter's eligibility, there remains a vulnerability wherein a dishonest authority could create false users, potentially manipulating the voting outcome to achieve a specific result. This vulnerability is present in many models proposed in the literature.

5.2.3 Khan et al. : voter identified by the hash of its credentials

Khan *et al.* proposed a model of voting system based on the blockchain platform Multichain, supposedly secure and E2E verifiable, in *Secure Digital Voting System Based on Blockchain Technology*[95], published in 2018. Their proposed model is based on the Prêt à Voter[116] approach, devised by Peter Ryan in 2009. The architecture of the model is presented with a layered approach, represented in Figure 5.1.

Layer 1	User Interaction and front-end Security
Layer 2	Access Control Management
Layer 3	e-Voting Transaction Management
Layer 4	Ledger Synchronization

Table 5.1: The layered architecture of the Khan's blockchain-based e-voting model [95].

The first layer is the User Interaction and Front-end Security layer which is responsible for interacting with the user and the administrator. It also handles the authentication phase, to retrieve the credentials that are required to communicate with the next layer. The Access Control Management layer defines the roles, access control policies, and voting transaction definitions. It verifies that the communications received at the frontend layer are allowed to be transferred to layer 3. The e-Voting Transaction Management layer maps the received votes onto the list of blockchain transactions to be mined. The credentials of the voter and the cast vote are used to generate a transaction ID. Pending transactions will be handled by mining nodes. The Ledger Synchronization layer synchronizes the data related to this specific application with the Multichain blockchain. Votes are also stored in a central backend database for further audit purposes.

The voting process is similar to many other models. First, the user logs in to the system. The final authentication method is left to the choice of the individual implementation of the proposed architecture. The paper suggests using classic username/password pairs, fingerprint, or iris recognition. The election authority stores in a database the fact that the user has received its credentials, in order not to allow

multiple votes from the same person. Once authenticated, the voter receives the list of candidates according to her constituency. The voter can then cast her vote, which is sent to the blockchain, as described in the architecture above. Once the transaction of the vote has been mined in a block, the user receives an email containing the transaction ID. Using this transaction ID, the voter can verify that her vote has been registered as cast. However, the transaction ID does not allow the extraction of information about the way the voter has voted.

Analysis of the model :
Regarding the *authentication* method, the paper suggests the use of fingerprints. The fingerprint scanned on the smartphone of the user would be sent to a server that will figure out if it matches the fingerprint of the citizen in the database. This solution implies multiple issues. First, it requires that the election's authority has a copy of all of its eligible citizens. This is not the case in most countries, including Belgium. In Belgium, the new eID cards contain an electronic copy of the owner's fingerprint, but these are not collected in a central database (see Section 3.5). Also, smartphones equipped with fingerprint readers do not send fingerprints over the internet. As a security measure, user's fingerprints are compared locally and stored securely and are not accessible by the OS or any application. Moreover, not every citizen owns a device equipped with a fingerprint reader.

The implementation ensures the principle of *one-human, one-vote* through the authority's storage of which voter have received their credential. However, a potential risk arises if a voter loses their credentials, for instance, by closing the browser window, and is unable to recover or obtain new credentials. The fact that the voter is unauthorized to get new credentials once it asked for them the first time, and not once it proceeds to vote for instance, is because the vote contains only the hash of the credentials, not the credentials themselves, preventing the authority from identifying the sender.

In this model, votes are not encrypted. The *privacy* of the voter is protected as she is only identified by the hash of her credential. With the collision resistance of the hash cryptographic function (at least in the random oracle model), the real identity cannot be retrieved except with an extremely low probability, and therefore the privacy of the user is protected. However, as the authority is delivering the credentials, the authority could re-compute the hash and therefore find the vote of the voter. This model requires trust in the authority, and being certain that the credentials will never be used or leaked. How to guarantee that authority cannot establish a link between voter and vote?

Receipt freeness is respected, as voters cannot prove how they voted. The transaction ID does not allow to identify the identity of the voter. However, the fact that a voter received the transaction ID is already a piece of information that can compromise secrecy. Also, the implementation of the model should ensure that the user shouldn't be able to prove their credentials belong to them, else the user could reproduce the hash of their credentials and prove their vote. *Integrity* is guaranteed thanks to the blockchain data structure.

On *verifiability*, individuals find themselves unable to verify that their vote is cast as intended due to the principles of receipt-freeness. However, they have the ability to verify that their vote is registered as cast individually. Also, anyone could verify that votes are counted as registered (in case the blockchain is public, which is not detailed in the model). Therefore the model is not completely end-to-end verifiable, contrarily to the claims of the authors.

In [78], the authors consider that the model is not secure, as it uses the Multichain framework, a private blockchain derived from Bitcoin susceptible to 51% attacks if more than half of the miners are dishonest. Also, the paper claims it is not enough *scalable* for nationwide elections. However, the Multichain framework can handle a throughput of up to a thousand transactions per second, making it one of the most scalable frameworks.

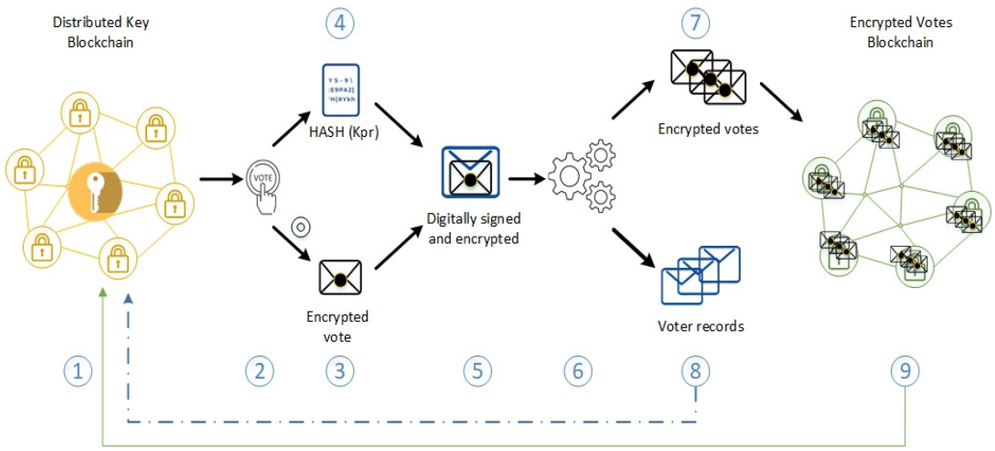


Figure 5.3: Proposed scheme from [103]

5.2.4 Using 2 separate blockchains to achieve anonymity and privacy

In the paper *Assuring Anonymity and Privacy in Electronic Voting with Distributed Technologies Based on Blockchain* [103] published in 2022, Neziri *et al.* propose an original method to guarantee both anonymity and privacy in blockchain-based voting system. They define privacy as «*when no one can know for whom and how the voter is voting, although the voter's identity is potentially known*»and anonymity as «*when no one knows for whom and how the voter voted, but it is potentially known what the voter is doing*». They claim that their model assures both privacy and anonymity. To do so, they are using two distinct blockchains: the Distributed Key Blockchain (DKB) and the Encrypted Votes Blockchain (EVB). The first will generate a set of public and private keys, while the second will store encrypted votes.

The scheme is depicted in Figure 5.3 and works as follows :

1. The DKB is responsible for the generation of the public keys that eligible voters will use to encrypt votes. Also, the DKB will verify the voter eligibility and previous voting status.
2. Upon the voter's request and consensus, the DKB generates the key pair for the specific eligible voter.
3. The voter encrypts the ballot and a cryptographic nonce using the generated public key : `E(V+nonce, Kpub(DKB))`. The nonce might be used later by the voter to verify her vote has been counted accurately.
4. The voter computes a hash of her generated private key: `HASH(KprVoter)`
5. The voter signs the encrypted vote (vote + nonce) and the hash of her private key. The hash allows the verification of the integrity of the vote.
   `SignKprVoter( HASH(KprVoter)+E(V+nonce, Kpub(DKB)))`
6. An anonymizer mixes timestamps and shuffles votes to reduce identification risks, ensuring voter data separation from the vote.
7. Encrypted votes are stored in the EVB after anonymization.
8. The voter's signature is removed from the encrypted ballot, ensuring the vote's unlinkability to the voter. Signatures are stored on the DKB to indicate whether a voter has voted or not, and to prevent double voting.
9. The EVB stores encrypted votes and the hash of the voter's private key.

At the end of the voting period, DKB will stop generating keys and the votes will be decrypted. This can be automatically managed using smart contracts. EVB signs the virtual ballot box, which contains the list of votes separated from voter information. DKB validates the signature using the EVB's public key and verifies that the number of votes received from EVB corresponds to the number of voter signatures. Then EVB decrypts the votes, tallies the votes, and publishes the results.

Analysis of the model :

This model of blockchain-based voting system uses an original method to guarantee

privacy and anonymity using two distinct blockchains to separate the keys of the voters and the encrypted votes. This paper is, to our knowledge, the only model that uses such architecture, even though Agora [8], analyzed below, also uses two distinct blockchains. We will analyze how each of the requirements is respected.

Integrity is maintained through a comparison between the total number of votes and the total number of signatures, preventing the addition of unauthorized votes. The use of a nonce enables each user to independently confirm that their vote has been accurately counted. Additionally, the hash of the private key ensures that the vote has not been altered. The system permits individual verification that the vote was cast as intended, recorded as cast, and global verification that the votes are counted as registered. It can therefore be considered *end-to-end verifiable*.

Receipt-freeness is maintained through the shuffling of encrypted votes and the separation of signatures from the remaining votes. This design prevents users from proving how they voted. Users could try to reproduce the computation if they retain the public key generated by the DKB and the nonce but there is no way to prove that this public key was indeed theirs, as the signature was separated from the vote.

Storing votes in the Encrypted Votes Blockchain (EVB) without attaching the voter's signature ensures both *anonymity* and *privacy*. Concurrently, recording the voter's signature in the Distributed Key Blockchain (DKB) helps prevent double voting.

The paper does not mention how the *eligibility* can be assured. We assume that it is the authority's responsibility to develop a protocol that authorizes the DKB to generate keys only for eligible voters that have been priory authenticated. The DKB is able to verify whether a voter has already cast a vote, preventing double voting and therefore assuring the *one-human, one-vote* property. On *scalability*, there is no maximum amount of users, as the authors do not provide information on the type of blockchain to use. They simply suggest using Ethereum as a public network or the Hyperledger platform as a permissionned blockchain network. The first is not scalable (at least considering the L1 of Ethereum), but the second can reach a high level of scalability. The author estimates that the size of one vote should not exceed 1 kilobyte, and therefore an election with 10 million voters would require a storage of 10 gigabytes per node, which is reasonable considering the storage space available on today's computers.

5.2.5 Arnab Mukherjee et al: Using ZKP for privacy in blockchain-based e-voting system

In the paper *A Privacy-Preserving Blockchain-based E-voting System*, published in 2023 by Arnab Mukherjee *et al.*, the authors propose a model of e-voting system based on blockchain that uses Zero-Knowledge proofs to guarantee the anonymity of the voters. The general idea is that only users who have registered using a ZK-proof procedure are allowed to vote, and their vote is anonymous as the only information revealed by the ZKP is that the user has the right to vote. Two distinct smart contracts are used: one that handles the whole process of registering the voters (`RegistrationSC`), and one that is responsible for all operations related to the voting process (`PollSC`).

The process described in the paper works as follows : First, all stakeholders register with an Identity Verifier V_j using Succinct Non-Interactive Arguments of Knowledge [110] (zk-SNARKs) in order to demonstrate the identity without disclosing any personal or sensitive information about the Prover. When a user (Prover) U_i wants to register, she sends to V_j a registration request Rq_i and a temporary key $TmpKey_i$. All communication are transmitted over a secure HTTPS channel to avoid attacks such as replay attacks. V_j sends an unverified identification called $UnvID_i$ to U_i. The Prover (U_i) then starts the Zero-Knowledge Proof procedure by determining a random number (r), later used to calculate a value (d), which is transmitted to the Verifier (V_j). V_j extracts d and sends two challenges to U_i. The Prover computes the answers (A_i) of the challenges and responds to V_j who will verify their validity. Once the verification is complete, a permanent identity ID_i is generated for user U_i. Registration and verifi-

cation reports are stored as documents using the InterPlanetary File System (IPFS), a decentralized peer-to-peer protocol designed to create a distributed method of storing and sharing files.

Once she has received her ID, any registered user can create a poll by calling the `createPoll()` method of `PollSC` with the necessary details (such as the name, the description, and the list of choices $PollOpts_{<x,i>}$). The ID of the Poll $Poll_x$ is returned. The user U_x who created the poll must then provide the list of registered users $[U_1, U_2, ..., U_j]$ who are allowed to vote to this $Poll_x$. Another solution is for the creator of the poll U_x to call the method `setOpen()` to make the $Poll_x$ open to all registered users on the system. The poll creator U_x can open the poll by calling the method `openPoll()`, and later call the method `closePoll()` to close it. During the period when the poll is open, registered users have the possibility to cast a vote invoking the method `castVote()` with parameters `PollID` and `PollChoiceIndex` among the options $< PollOpt_1, PollOpt_2, ...PollOpt_y >$. `PollSC` verifies that the user is registered and eligible for the vote (*i.e.* she is either in list $[U_1, U_2, ..., U_j]$ or the poll is `setOpen()` to every registered user). At then end, U_x can close the poll. Votes cast when the poll is closed will not be counted. Once the poll is closed, any registered stakeholder can call `getPollResults()` to receive the results of the poll. If the poll was not closed yet, this method will return `null`.

The authors have tested their model on the Ethereum testnet Rinkeby, with the two smart contracts written in Solidity. The Zero-Knowledge Proofs are created and validated inside the smart contracts and developed using ZoKrates, a toolbox for implementing zk-SNARK on Ethereum.

They conducted a performance analysis of simulating computations on these smart contracts and presented data on gas costs and system latency. On gas cost first, data shows the gas unit consumed for each method call, which varies between 7000 units (`closePoll()`) and 213.000 units (`createPoll()`). The entire process to register one user is around 475.000 units, the management of one voting poll consumes in total 450.000 units for the poll creator, and casting one vote for one registered user costs 19.000 units. The final cost of these operations on the Ethereum network would depend on the gas price at the time of each operation. The dynamics of the fees in a permissionless blockchain are explained in Section 3.2.6. With a gas price ranging between 20 and 100Gwei (it may increase up to hundreds of Gwei during peak activity), we can take the high estimation of 100 Gwei, and a priority fee of 2, and an ETH price of 2000 USD. The total price is given by this formula :

$$Transaction fee = Gas limit * (Base fee + Priority fee) * 10^{-9} ETH/gwei * ETH price in \$$$

$$= 450000 * (100 + 2) * 0.000000001 * 2000 = 0.0459 ETH * 2000 USD/ETH = 91.8\$$$

The cost of managing a poll is, at the time of writing, around 90 dollars. The registration of each voter also costs around the same price, and casting a vote costs around 4 dollars. Regarding latency, the paper shows that read operations have a constant cost (as it does not require any modification of the blockchain), but the latency of write operations (such as casting a vote) increases logarithmically with the number of transactions per second.

Analysis of this model :
This model is very similar to the naïve example described above (Section 5.1) or to the Hjálmarsson model (Section 5.2.1). The similarities remain in the aspect that a smart contract (here, the `PollSC`) handles the voting process and keeps track of all the unencrypted votes. In both cases, a cast vote simply contains the ID of the voter and the choice of the voter (and additionally the `PollID` in this model). The major difference with these previous models is that here, the ID of the voter is obtained after the authentication with the ZKP, so even the election authority cannot establish any link between its ID and its real identity.

As any user can create a poll, this model is not made specifically for an election (as the paper suggested), but instead for any kind of voting process where someone or some

institution wants to proceed to a vote. The creator of the Poll Event must provide a list of authorized voters. In the case of an official election, the election authority must create the Poll Event, transmit the PollID (which will be the only official poll to use), and provide the list of all eligible users who have authenticated using ZKP.

As shown by the results of the experiment, the high transaction fees make it unrealizable for an election, as per definition the voting process must be free for every voter. When the price of one `ether`, the native token of the Ethereum network, was around 10$, or when the gas fees were lower (due to a lower activity on the network), the price could have been relatively acceptable. The current market price and congestion of Ethereum make it unsuitable for an election. Also, the fact that the gas fees change with time and with the market price of the currency is also something not imaginable for an official election. These fees are one of the problems with the use of public blockchain. Also, the use of Ethereum does not permit sufficient scalability. However, the use of a private or permissioned blockchain might solve these issues.

Regarding the requirement of *anonymity* and *authenticity*, this model succeeds thanks to its use of Zero Knowledge Proof. Anonymity of the voter is indeed guaranteed as even the election authority cannot link the ID of a voter with its real identity. However, the application of ZKP for *authentication* presents a double-edged sword. While it effectively guarantees privacy, it also poses challenges for the election authority in auditing voter authentication. In the event of a failure in the ZKP process, such as insufficient proof of a voter's *eligibility*, there is no recourse to correct the issue and eliminate invalid votes. Also, the election authority will not be able to know whether an eligible voter has voted or not, which might be problematic in countries with mandatory vote. The requirement of one *human, one vote* (*un-reusability*) can be achieved by assuring in the ZKP procedure that the voter is not already registered. Regarding the requirement of *availability*, this model indeed implements a feature of opening and closing the poll with the methods `openPoll()` and `closePoll()`. Note that any node executing the smart contract could get the results computed on the fly, even if the poll is still open.

Regarding the other requirements, this model does not offer further improvements. For instance, *integrity* is simply protected by the properties of the blockchain.

The paper does not provide information regarding the *verifiability* of individual voters and simply argues that auditors can verify the correct computation of the results. If the blockchain is public, then the voter could verify that her vote is well-registered, by accessing to the vote corresponding to the ID received after the ZKP authentication. The auditors, and any other observer, can verify that all registered votes are tallied. This process is straightforward, as the votes are not encrypted. The model is therefore anonymity *end-to-end verifiable*. However, the *receipt-freeness* is not respected. By reproducing the ZKP, or disclosing the received anonymous ID, the voter can then reveal its votes, as those are stored unencrypted.

In conclusion, this model was not thought to be used for election specifically. If implemented on a public blockchain, it implies problems such as a severe limitation in scalability, variable transaction fees, and all the security issues that might be related (such as a 51% attack). Those problems might be solved with the use of a private or permissioned blockchain. The strength of the model is the protection of anonymity with the use of Zero-Knowledge Proof (more specifically zk-SNARKS), but it prevents any attempts for the election authority to know who has voted, and to later revoke invalid votes. Therefore the ZKP procedure must be meticulously designed to avoid any error. As the votes are not encrypted and are simply linked to anonymous IDs, it offers straightforward transparency but not receipt-freeness nor coercion-resistance.

5.2.6 Agora

Founded in 2015, Agora is a commercial entity specializing in blockchain-based voting technology. The company provides services to institutions and governments for organizing secure elections. Detailed extensively in a 40-page whitepaper [8], Agora's model encompasses a permissioned blockchain for managing the voting process and utilizes

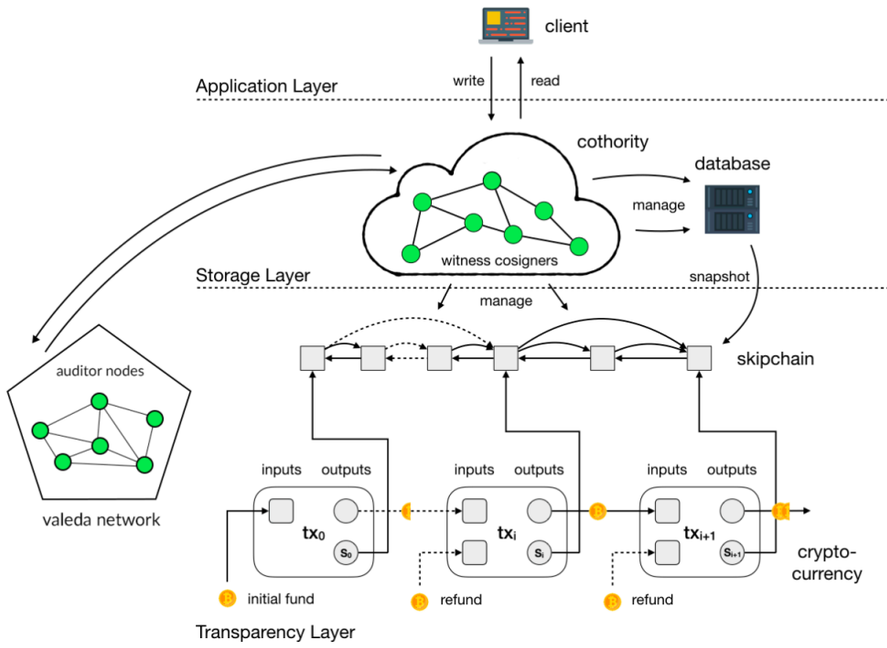


Figure 5.4: Agora's multi-layers Architecture. From [8]

the Bitcoin blockchain for log storage, leveraging its security and decentralization. The system's transparency is assured as all steps are auditable. Agora's architecture is built on a multi-layered framework including the Bulletin Board, Cotena, Valeda, and user applications like Votapp, depicted in Figure 5.4. All of these components are detailed hereafter.

Multi-layered architecture

The Bulletin Board, serving as Agora's system core, is a blockchain managed by write-permissioned nodes, maintained by Agora and authorized partners, known collectively as the Cothority, while offering public read-only access to ensure transparency. It records all the data during the election process (encrypted votes, zero-knowledge proofs, anonymized decrypted votes, results, etc). The data structure of the Bulletin Board is a skip list, incorporating not only sequential links but also additional forward and backward links. These links allow for efficient navigation across the blockchain, instead of linear traversal.

All of the Consensus Nodes, forming together the Cothority, maintain a complete copy of all transactions on the network. Among these nodes, one is designated as an 'oracle node' on a rotating basis. This oracle node assumes a pivotal role in the network's functioning: it receives ballots and other data, proposes new blocks to the network, and is responsible for writing confirmed blocks to the Cotena log. The nodes independently monitor each other, ensuring an unaltered and accurate record of the system's data. They maintain the bulletin board, authenticate the ballots, confirm the blocks proposed by the oracle node, verify the correctness of the Cotena log, and, after the election, decrypt the anonymized ballots.

Cotena is the ecosystem's second layer, whose purpose is to store tamper-resistant logging. It stores cryptographic proofs of the Skipchain (Agora's custom blockchain) on the Bitcoin blockchain. This linkage ensures that the data on Agora's Bulletin Board is not only secure within its own ecosystem but also benefits from the robustness and widespread trust of the Bitcoin network. Cotena operates by keeping a record of specific Bitcoin transactions. These transactions contain application-specific statements, implemented using Bitcoin's OP_RETURN opcode. This method requires downloading only a limited set of data from the Bitcoin blockchain, which includes all block headers (80 bytes each), the Merkle tree, and only those Bitcoin transactions that are relevant to Agora's system. This data takes in a total of around 350 MB of memory

size, which is much more convenient than maintaining the entire Bitcoin blockchain which, at the time of writing, represents 530 GB.

Valeda is a decentralized network of trustless nodes responsible for validating election results recorded on the Bulletin Board, playing the role of the auditors. Operating an Auditor Node in Valeda requires completing KYC (Know Your Customer) procedures and token staking, ensuring network participant commitment and authenticity.

Agora provides an official application, Voteapp, but the system's open architecture allows for the development of third-party, user-friendly applications interacting with the Bulletin Board. Votapp comprises three main components: Voting Booth, the app used by the user to cast and verify their vote, an Audit App to enable authorized users to participate in the Valeda network, and a Node App that allows any individual to run a read-only node.

Agora's voting process

The voting process follows the classical timeline of most e-voting model: configuration, casting, anonymization, decryption, tallying, and auditing. During the configuration phase, the election administrator creates an election event, specifying in a configuration file the public keys of election officials, the election parameters, the starting and ending times, the list of candidates or choices/options, and the list of all eligible voters for the given election. This configuration file is then hashed (this hash becomes the ID of the file), signed by the administrators, and then published on the Bulletin Board. During the voting period, each eligible voter can submit her vote using the Voteapp voting both, any third-party app interacting with the bulletin board, or even with a compatible voting machine in the voting station, in the case of a hybrid on-site/online election. The user first authenticate, then select the candidate of her choice, and then validate. The vote is then encrypted with threshold ElGamal encryption using the Cothority's collective distributed public key. The user has the possibility to verify that her vote was cast-as-intended by using a second device that will recompute indecently the encryption and should return the same ciphertext. If a sufficient number of voters perform this validation and no error is found, then the probability that the vote casting is not done properly becomes negligible. Once the voter has validated her choice, and had the possibility to verify the encryption, the encrypted vote is signed with the voter's digital identity credentials and sent to one of the nodes of the Cothority network. The Cothority network will authenticate the vote, and if valid, will add it to the Bulletin Board blockchain.

To preserve anonymity in Agora's voting system, all encrypted ballots are sent through a Neff-shuffle-based [102] mixing network. In this network, mixing nodes process the entire list of encrypted ballots and output a new list of anonymized ballots. Each node provides zero-knowledge proofs of proper shuffling to the Bulletin Board, ensuring the correctness and auditability of the anonymization process

To perform tallying, Cothority nodes first verify the zero-knowledge proofs from the anonymization phase. After validation, they collectively decrypt the anonymized ballots. Each node partially decrypts the votes, creating a zero-knowledge proof for each partial decryption's correctness. These results are then published on the Bulletin Board. Election administrators confirm the validity of these proofs, and if a sufficient number pass validation, they use the partially decrypted ballots to reconstruct the original plaintext ballots. Finally, these plaintext ballots are posted on the Bulletin Board for tallying. The administrator tallies the anonymized plaintext ballots, signs the results, and publishes them on the Bulletin Board.

As all the data and the cryptographic proofs are stored on the Bulletin Boards, auditors can verify each step of the process.

Analysis of this model :

The Agora blockchain-based e-voting system's model appears quite promising, with a detailed and comprehensive whitepaper describing each step of the process.

In terms of *authenticity*, the document indicates that Agora depends on election administrators to choose an appropriate identity management system and to implement voter authentication mechanisms. However, the paper lacks specifics on voter identification within the system, such as the method of signing and the format of the voter eligibility list. It does not clarify whether the list includes personal identifiers like real names, social security numbers, or ID numbers. Using real names could raise concerns about personal data protection, while arbitrary ID tokens might turn out to be insecure. In Belgium, a potential solution could involve the utilization of private and public keys embedded in Belgian eID cards. The list could include public keys of eligible voters, and the voters would later sign with their private key which remains within the card's chip, thus ensuring both privacy and security. As encrypted votes are first authenticated by the Cothority, and then anonymized through the mixing network accompanied by Zero-Knowledge Proofs, it ensures that only authenticated votes remain in the final output.

Concerning *integrity*, the model gets robustness from its own permissioned blockchain and further reinforces security by archiving logs and snapshots on the Bitcoin network. However, relying on Bitcoin's network for added security might imply a lack of confidence in its own system's *security*. Questions also arise regarding the selection and trustworthiness of Agora's "partners" who run nodes that are part of the Cothority.

Regarding *availability*, the starting and ending periods are defined in the configuration file of the election event. Additionally, due to the geographical distribution of nodes, the risk of a Denial of Service (DoS) attack is minimized.

On *anonymity*, the system performs well thanks to the implementation of a mixer network, which effectively ensures that there is no possibility to link votes to individual voters, even for Agora or the election authority.

The Agora system is designed to be *end-to-end verifiable*, allowing for independent verification of each stage in the election process. According to its whitepaper, the system claims to ensure both cast-as-intended verification and adherence to receipt-freeness. It enables users to confirm that their vote has been cast as intended through independent recomputation of the encryption. However, it raises the question of whether this process amounts to creating a "receipt" of the vote, potentially contradicting the principle of receipt-freeness. Unfortunately, the whitepaper does not provide detailed insights into how the system simultaneously maintains receipt-freeness while ensuring end-to-end verifiability. The ZKP resulting from the shuffling ensure the registered-as-cast property, and the ZKP resulting from the decryption ensures the tallied-as-registered properties. Therefore the model is indeed end-to-end verifiable.

The *auditability* is placed at the center of the ecosystem. The Valeda network, composed of Auditor Nodes, inspects the voting process all along. Moreover, anyone (after a simple KYC procedure) can run a read-only node to maintain a copy of the Bulletin Board and reproduce the computation. Each step is auditable and verifiable thanks to the use of ZKP.

On paper, Agora checks all the boxes on the generic requirements of an election.

The business model of Agora is to sell their native tokens VOTE (a cryptocurrency) which is used to incentivize and compensate participants in the election process. Governments should buy tokens to organize an election on the Agora network, supplying funds for the project development and increase the token price. This funding method allows the voters to vote for free, with only the election organizers who pay. However, Ahmed Yusuf Patel *et al.* argue that proprietary blockchains are not suitable for online voting [107] and prefer largely open-source blockchains to enhance security, transparency and accessibility. A closed-source solution does not allow flexibility for countries to adapt the program to their need. This business model does not seem to have worked, since the development seems to have stopped in 2020.

Agora made the headlines [111] in 2018 when the company was involved in the Presidential Election in Sierra Leone, considered by many (*e.g.* [82, 138]) as «*The First Blockchain-Backed Presidential Election in the World*». However, this statement

was exaggerated. Based on Agora's official statement [130], the company participated as an accredited international observer in the Sierra Leone election, covering 280 polling stations in the country's West Districts. They partially implemented their technology during this electoral process, recording the tallying results on the blockchain, but the voting process in itself was done on paper in the most traditional manner.

5.3 Conclusion of the literature review

The six models analyzed above present distinct architectural designs. We will summarize these analyses by comparing how each model addresses each requirement.

Authenticity/Eligibility: The responsibility for verifying voter authenticity and eligibility lies with the election authority for every model. All models enforce the principle of *unreusability*, ensuring each eligible voter casts only one vote. The authority's role includes compiling the electoral list and ensuring only eligible voters are authenticated once.

Integrity: All models rely on the blockchain's inherent tamper-proof nature to ensure ballot integrity, operating under the assumption that if blockchain-stored votes are immutable, then they are inherently valid. Hjálmarsson, Kirillov, Khan, and Agora use permissioned blockchain, to prevent an external actor from taking control of the network. Additional safeguards are integrated in some approaches, like Neziri's dual blockchain system, which verifies that the count of signatures in the Distributed Key Blockchain (DKB) is equal to the number of votes in the Encrypted Votes Blockchain (EVB). Similarly, Agora enhances its custom blockchain's security by also transmitting snapshots of its Bulletin Board to the Bitcoin blockchain.

Anonymity/privacy: Different methodologies are employed across different models to address anonymity. Hjálmarsson's approach, which uses smart contracts and unencrypted votes, requires the nodes to remove the voter's addresses and signatures, resulting in voters' identities being secretly disclosed along with their votes to the nodes running the permissionned network, and diminishing transparency. This is an unacceptable risk regarding vote secrecy. Mukherjee's model also utilizes smart contracts with unencrypted votes, but assigns an anonymous token ID, obtained through Zero-Knowledge Proofs (ZKP), to each voter. This token confirms voter eligibility without revealing personal data. Khan's method involves using the hash of the voter's credentials for anonymity. Neziri shuffles the votes to separate signatures and encrypted votes. This method is close to the Identity Mixers (idemix), used by Agora to separate voter identities and votes while maintaining correctness via ZKPs, similar to the iVote system in New South Wales. Kirillov uses Idemix in addition to blind signatures, to prove their eligibility by registering anonymously their public key.

Verifiability : Neziri, Mukherjee, and Agora offer *end-to-end verifiability*, meaning that it allows individual verification that the votes are cast-as-intended and registered-as-cast, and public verification that all votes are tallied-as-registered. Hjálmarsson allows individuals to verify that their vote is registered and counted, but do not allow public verification, resulting in a lack of transparency. Kirillov allows voters to verify their vote was counted, and auditor nodes can verify the conformity of ZKPs. Khan model allows public verification that votes are tallied as registered and allows users to verify that their vote was registered as cast, but does not allow individual verification that the vote was cast as intended to preserve receipt-freeness.

Receipt-freeness and coercion resistance: Receipt-freeness is often viewed as a desirable feature for electronic voting systems to resist coercion. Some online voting models that lack this feature enable voters to change their vote later to reduce coercion risks. Agora and Neziri's dual blockchain model both achieve end-to-end verifiability and receipt-freeness, thanks to their sophisticated anonymization processes. Khan's model also maintains receipt-freeness, as voters receive a transaction ID by mail to confirm their vote's inclusion, but this ID doesn't reveal the vote's content. Kirillov's model, prioritizing verification over receipt-freeness, allows vote modification by revoking and regenerating keys, thereby mitigating coercion. Hjálmarsson's model permits voters to check their vote's content, potentially serving as a receipt. The

inability to change the vote post-casting offers no coercion resistance.

Availability/Tallying period: Usually, the results are published at the end of the election to prevent it from influencing participation. The models using post-vote anonymization methods (e.g. mixing networks) like Agora, Neziri's, and Kirillov's decrypt the votes at the end of the voting phase, so the results cannot be revealed earlier. In Hjálmarsson's and Khan's models, the tally is done on the fly by the Smart Contract, therefore revealing it in real time publicly (an alternative is to run the blockchain on a private network but at the drawback of losing transparency). This is also the case in Mukherjee's model, but the Smart Contract revealing the results can only be called at the end of the voting phase. However, any entity running a node could compute and display the results at all times.

Type of blockchain and scalability : Mukherjee designed its model for Ethereum and tested it on the Ethereum testnet Rinkeby, a permissionless blockchain compatible with EVM. However, opting for a permissionless blockchain leads to limitations such as reduced transaction throughput and the requirement of gas fees. This results in higher costs per voter and fluctuating prices dependent on market conditions. To prevent these issues, Hjálmarsson suggests using a Go-Ethereum framework on a permissionned blockchain, to enjoy the use of Ethereum Virtual Machine (EVM) on a permissionned and therefore more scalable blockchain. Khan proposes using the Multichain framework, which supports a theoretical throughput of up to 1000 transactions per second. Kirillov developed its model on Hyperledger Fabric, a framework reputed for being highly scalable and supporting a high transaction rate. However, the authors themselves consider Hyperledger Fabric to be insufficiently scalable for large-scale national elections in [85]. Agora has developed its own proprietary custom blockchain and claims scalability in its whitepaper. However, no data is provided to confirm or refute these claims. In a literature review [78] comparing different commercial solutions of blockchain-based voting systems including Agora, the authors concluded that no solution was scaling enough for national elections. Finally, Neziri does not provide information considering what type of blockchain is used in its model. It is therefore not possible to estimate its scalability.

Figure 5.5 illustrates the performance of each model against each requirement previously compared, rated on a scale from "good" (1) to "bad" (5), based on estimations rather than scientific calculations. Where these values are not determinable, the corresponding cell is left blank. The criteria of authenticity/eligibility and unreusability (one-human, one-vote) are not included in this representation, as they are the same for all models with the responsibility placed on the election authority.

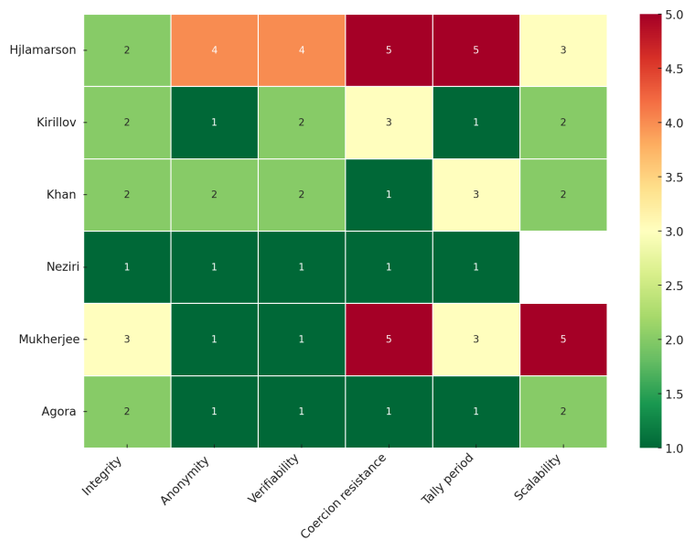


Figure 5.5: Visual representation of the performance of each model against some comparison criteria. Values are estimated on an approximated scale from "good" (1) to "bad" (5).

Chapter 6

Proposed solution

We will now present a blockchain-based online voting system that could be used for elections in Belgium. We will first motivate our design choice for the use of each method, tool, or technology. Then, we will provide an extensive description of the proposed solution. In the next chapter 7.3 we will analyze the performance of the model regarding the requirements defined in Section 4.1, and we will consider the non-technical aspects, such as legal considerations, and societal impacts.

6.1 Selection of Blockchain type and framework

The first design choice concerns the selection of the network type (public/private, permissioned/permissionless) for our blockchain-based i-voting system. Let's start with a reminder of the motivation to use blockchain in the first place: preventing potential attackers from manipulating votes on a central server, preventing dishonest election organizers or system administrators from vote tampering, enhancing robustness through multiple nodes to avoid a single point of failure, and ensuring transparency by enabling public verification of tally correctness.

In balancing security, scalability, and decentralization, as discussed in Section 3.2.9, high-stakes elections necessitate prioritizing security. As discussed in Section 4.3.2, the system must handle millions of transactions within a limited timeframe of a few hours, requiring a significantly important scalability. Contrary to the total decentralization seen in permissionless networks like Bitcoin and Ethereum, our system does not require such an extensive spread. As previously mentioned, the goal is to have multiple nodes to distribute responsibility and minimize the likelihood of a successful cyberattack on a centralized server. A limited number of highly trusted nodes is preferable to ensure security and reliability, rather than a vast, permissionless network where external entities might gain undue control, such as in a 51% attack. For these reasons, we are opting for a permissioned network composed of a list of nodes selected prior to the election.

We opted for the Hyperledger Fabric framework to manage our blockchain network. Hyperledger Fabric, introduced in Section 3.3, presents multiple advantages. First, it proposes a modular architecture, enabling a highly customizable permissioned network where organizations and peers can be defined with rights and roles. Moreover, it includes a feature of private data collection, enabling the storage of private data (such as some private keys), accessible only by authorized peers and organizations. Finally, Hyperledger Fabric is, at the time of writing, one of the state-of-the-art solutions in terms of scalability, handling up to thousands of transactions per second. Despite the criticisms expressed in [85] regarding Fabric's scalability for e-voting, we consider that this current scalability is sufficient considering the capacity requirements defined in Section 4.3.2 and that future works on Fabrics might continue to improve its scalability. Moreover, in Fabric, every channel functions as an independent ledger. This feature allows for the possibility of creating a distinct channel for each province and aggregating the final results. By doing so, the workload is distributed, as each channel operates concurrently. This approach not only reduces the load on each individual channel but also enhances the overall scalability.

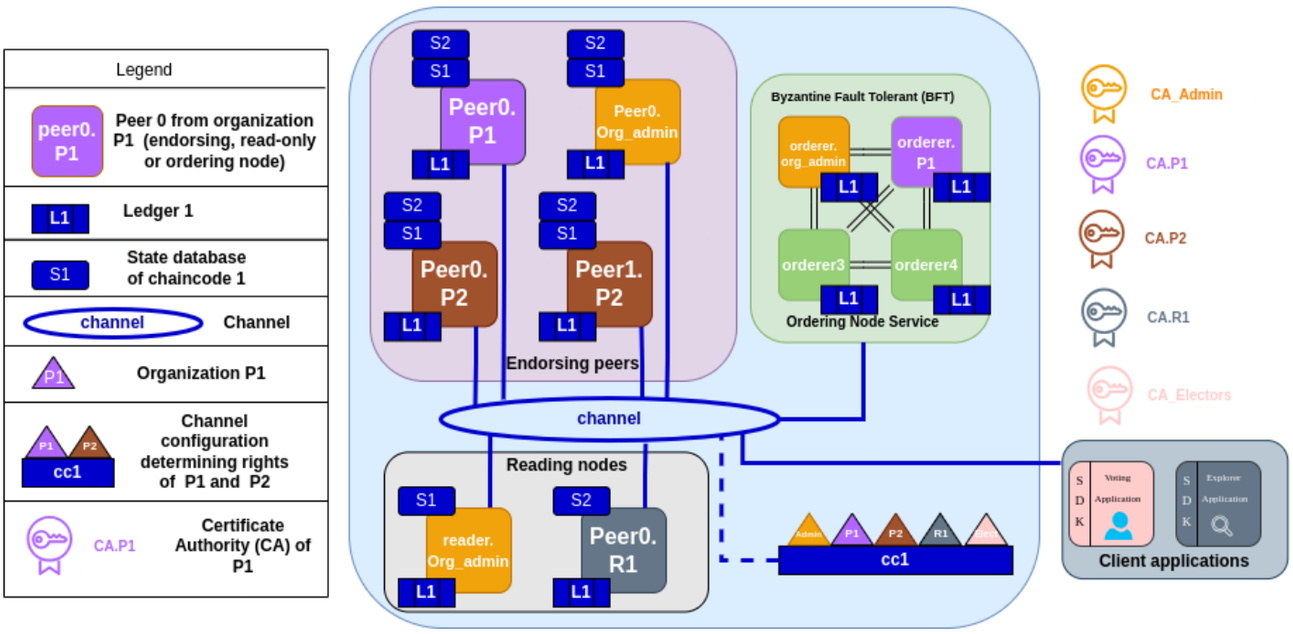


Figure 6.1: The architecture of the Fabric network in the proposed solution.

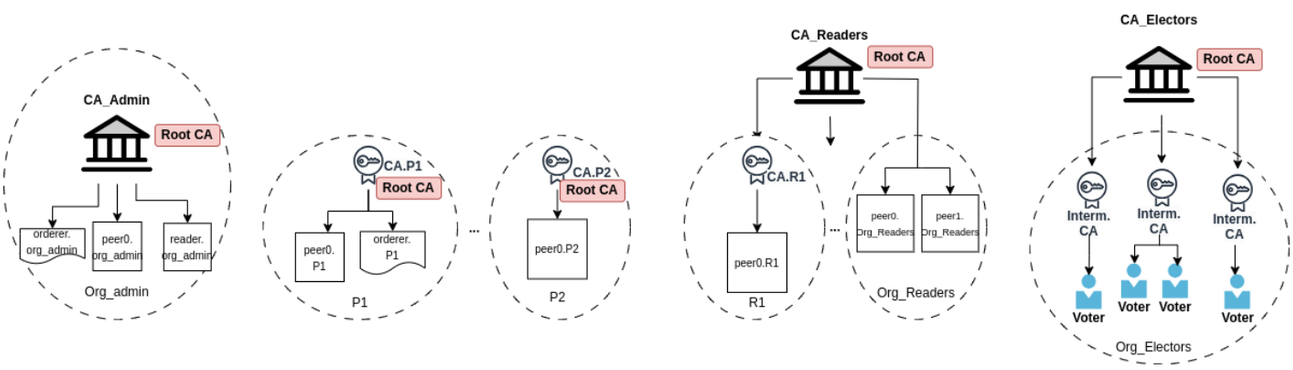


Figure 6.2: A diagram representing organizations and their Certificate Authority inside the network. From left to right: the Admin organization responsible for the election management, the Partner organizations running ordering nodes and peers, the Reader organizations running read-only nodes, and the set of voters included into one big organization *Org_electors*.

6.2 Network's organization

In this section, we propose a Fabric network configuration including endorsing peers and ordering and reading nodes, client applications, and Certificate Authorities. For a better understanding, we recommend the reader first to be familiar with the concepts detailed in Section 3.3. Figure 6.1 depicts a simplified version of the network in our solution with few organizations, for clarity reasons. The Fabric framework is very modular, and this model is designed to be flexible, simplifying the management of stakeholder addition or revocation. If this model ever had to be implemented, the precise list of stakeholders should be defined based on security analysis.

Organizations and Certificate Authority

The Fabric *network* includes all the organizations, peers, ordering nodes, channels, smart contracts, and other components involved in the election process. Remind that any device (node or client app) must be part of an authorized organization and be authenticated with a certificate issued by a trusted CA.
Different organizations take part in the network. Figure 6.2 depicts the different types of organizations involved, the nodes each of them runs, and the Chain of Trust formed by the CAs.

- The **election administrators** *Org_admin*, who will create the channels and

manage the elections. The organization has its own Certification Authority CA_admin. This organization can also run peers.

- **Partners organizations** P_1, P_2, .., P_n are all the institutions running one or multiple nodes in the permissionned network. These can include public institutions with cybersecurity expertise, universities, private companies specialized in cybersecurity, Cloud providers, etc. Each of these organizations has its own CA_P_i to authenticate its peers. Between ten and twenty organizations should be a good order of magnitude. All of these organizations are independent in the network and recognized as administrators in the channel configuration. This means that any change in the channel policy must be approved by (at least the majority of) these organizations. An alternative would be that all the Partners CAs are themselves authenticated by a Root CA called $CA\_Partners$ managed by the election authority. While this second solution facilitates network management to easily add and revoke organizations, the first one is more resilient in case of CA corruption. The organizations running a peer must be carefully selected, as they run the few peers managing the permissioned network.
- **Reader organizations** R_1, R_2, ..., R_n contain peers with read-only rights, watching the network for transparency. These peers could be managed by or for auditors, NGOs, journalists, or curious citizens. As Hyperledger is a permissioned network with only authorized nodes able to interact, these peers must also be authenticated and included in organizations. These organizations can independently manage their nodes through their own CAs, authenticated initially by CA_Reader, another Root CA managed by the election authority. Alternatively, peers can be part of a larger organization $Org_Readers$ grouping all "single reader-nodes" directly authenticated by $CA\_Readers$, in order to facilitate the deployment of such read node.
  The process of requesting a certificate for a read-only peer should be much easier than the one to request a committing node, as a compromised read-only node will not affect the network security. This process could be a simple KYC (Know Your Customer) procedure, as applied in Agora's Valeda network, a set of nodes that serve audit purposes. In case of spamming/DoS attack, the Root CA can simply revoke the certificate. New read-only peers can therefore easily be added to the network, even when the election has already started.
- The set of all the eligible **electors** forms one big organization $Org\_Electors$. The CA charged for the authentication of all voters is called $CA\_Electors$ and is managed by the election authority. In practice, this CA can be divided into one Root CA and multiple Intermediate CAs. The process in which voters authenticate and receive their certificates is detailed later in this section.

Membership Service Provider (MSP)

In Hyperledger Fabric, MSPs (Membership Service Providers) define the permissions and roles within a channel. The Certificate Authority (CA) is responsible for identifying channel members but does not offer further details beyond confirming their recognition by trusted entities. For instance, Reader organizations are granted only read access, unable to endorse transactions or order blocks. Partner organizations manage their own CAs, allowing flexibility in peer management, even adding new peers mid-process. However, MSPs restrict the number of peers per organization to prevent any single entity from dominating the ordering service.

Consensus Protocol

The ordering nodes will use the Byzantine Fault Tolerance (BFT) consensus protocol, supported by Hyperledger Fabric since version 3.0 and described in Section 3.3. BFT ensures system resilience against both standard failures and malicious attacks, functioning correctly as long as at least two-thirds of the nodes (plus one) remain honest.

In the event of a cyberattack on one of the ordering nodes, the system will re-

main operational, provided that less than a third of the nodes are compromised. The selection of ordering nodes from Partner organizations with computing resources and high-security standards makes a widespread attack challenging. In scenarios where a node is compromised and acts maliciously, it remains possible to revoke its privileges. Depending on the architectural choice made regarding the Certificate Authority of the Partner organizations, this can be achieved either by $CA\_Partners$ revoking the malfunctioning organization's certificate or through a majority agreement among the remaining organizations to remove the dysfunctional entity from the channel, as stipulated by the channel policy.

Channels and chaincodes

Within the Fabric network, the Org_admin creates one or several channels. An entire election can be managed within a single channel, or it can be segmented across multiple channels. Each channel can establish its own permissions. For instance, Belgian regions or provinces may prefer separate channels for some reason (e.g. defining their own list of Partner organizations). Splitting into multiple channels reduces workload per channel, but the ordering nodes then have to independently manage the consensus algorithm for each channel. Please remember from Section 3.3 that a peer can belong to multiple channels and therefore host multiple ledgers and that each ledger can have multiple chaincodes that interact with it.

In Belgian elections, the available candidate list varies by the voter's residency and election type. For federal elections, a Walloon voter, for example, can only vote for French-speaking parties. A part of the territory where all citizens can vote for the same candidates in an election is called an electoral district or constituency (*"circonscription électorale"* in French). There are different constituencies for elections to the House of Representatives, the parliaments of the Regions and Communities, and the European Parliament [41]. The size of a constituency can range from a province (or the Brussels-Capital Region) for elections to the House of Representatives for instance, to a municipality for municipal elections. As we want our model to work for every type of election in Belgium, we will refer hereafter to these districts as their smallest possibility: the municipalities. However, keep in mind that the election configuration could be easily adapted for elections with bigger districts.

Accordingly, a smart contract (chaincode) is deployed for each municipality and for each election happening on the same day. A voter is assigned to their municipality of residence, as indicated in the OU (organization unit) field of their certificate. The Attribute-Based Access Control (ABAC) of the chaincode ensures that only a voter with the correct `OU` attribute is allowed to interact with it. Each chaincode has a pair of private/public keys, used for the generation of blind signatures. The private keys are stored in the private data collection, and only accessible to authorized peers. They are not transmitted to the ordering nodes.

In conclusion, the HLF network for election can be divided into channels, and these channels may contain multiple chaincode, for instance, one per municipality and per election. The peers endorsing transactions to their attributed chaincodes store privately the associate private keys.

Client applications

In Fabric, applications interact with chaincodes via the Hyperledger Fabric Gateway featured in the SDK. The Gateway acts as an interface between the client and the nodes, managing the transmission of the flow of transaction proposals and their response. As previously mentioned, these clients need to be authenticated to interact with the network, just as peers do. Each authenticated user operates a client app to interact with the chaincode, using a certificate from CA_Elect, and can create transaction proposals. Reader nodes utilize their applications for purposes like blockchain exploration, enabling public access to blockchain data, or real-time election monitoring, with read-only access authenticated by $CA\_Readers$. Administrators and partner organizations may also develop apps for efficient peer configuration. Once again,

they must be authenticated by their respective organizational CAs with corresponding rights.

The role for the election authority

While this model decentralizes responsibility across multiple nodes and organizations rather than a single government-run server, election authorities retain significant roles in the process. They act as election administrators (Org_admin), manage the Root CA for Readers organizations ($CA\_Readers$), handle the Authentication Service (see later), and generate voter certificates ($CA\_Electors$). They may also operate their own peers and ordering nodes (`peer0.Org_admin, orderer.Org_admin`) that participate in the consensus and reader node for auditing or public data sharing. Moreover, some Partner organizations can be controlled by the government (like public institutions), or related to it. To mitigate centralization risks, dividing roles among different independent institutions, regulating processes through procedures and laws, and distributing keys among multiple participants are suggested best practices. This is also a reason why the Partners CAs should be directly recognized in the channels policies, and not authenticated by another CA run by the election authority. Otherwise, by revoking all certificates, the election authority could take ownership of the entire network for itself.

6.3 Authentication of the voter

Every model analyzed in Section 5 depended on an external authentication process between the voter and the election authority, but no additional information was provided. This paper proposes an identification method that could be integrated into the Belgian electoral system. Two elements have to be taken into consideration: Firstly, the elector must authenticate their identity to the authority by proving her real identity. Secondly, due to the architecture of Fabric, any network participant, whether interacting with peers or smart contracts, must be authenticated with certificates issued by a recognized Certificate Authority (CA). Therefore, the authentication protocol we propose necessitates the user to establish their real identity and results in the issuance of a certificate by a CA.

We take advantage of the digital tools already existing in Belgium for the authentication steps by using the digital identity app itsme® presented in Section 3.5. For those unable or unwilling to use itsme® but still desiring to participate in online voting, alternatives could include authentication in person at an administrative office, such as City Hall, or utilizing other methods provided by CSAM (the Belgian service for logging onto online public services), such as a Belgian eID with a card reader (see Section 3.5) or receiving a code via SMS. CSAM also allows citizen from EU countries to authenticate to Belgian services using digital keys emitted by their origin country. These alternatives, however, are not the focus of this document. Given that our model permits voters the choice between online and on-site voting, it can be posited that voters not opting for itsme® can simply resort to the latter option and vote on-site. In the future, another possible alternative authentication method could be the European Digital Identity Wallet (see Section 3.5) which is still in pilot testing at the time of writing.

Storing the voter list

Just like in regular elections, the authorities must establish a list of registered eligible voters. Unregistered individuals desiring to participate in the voting process have to request their inclusion on this list. EU citizens who have resided in Belgium for a minimum of five years have the right to be registered for local communal elections. In the proposed online voting system, voter authentication is also conducted online, necessitating secure database storage of the voter list and a mechanism to prevent double voting.

It is worth noting that the voter list is accessible to citizens upon request and political parties receive two copies [41]. However, it is forbidden to share or publish publicly this list. Therefore, the list of eligible voters cannot be stored on the blockchain (which would have been good for transparency but not compliant with regulation), and the system relies on a centralized model for the authentication part. Independent auditing is necessary to ensure that the election authority does not issue false certificates to non-existent, absentee, or deceased voters and respects un-reusability.

Authentication using itsme®

On the website or the mobile app used to vote online, the first step will be to authenticate to an Authentication Service (AS). After a short text explanation of how the authentication works and a small video illustrating the following steps, the user clicks on an orange button "Authenticate with itsme®". To simplify the voting process on voting day and to lower the workload on the itsme® and the AS servers, it should be recommended that voters create their itsme® account before the election. But if the user does still not have an itsme® account, she is invited to install the app and create an account, proving her identity with her eID and a card reader, or with a bank card and the associated card reader. A technical description of the authentication with itsme® is detailed in the official documentation [77]. Here are the main steps involved.
If the connection is initiated **from a web browser** on a computer :

a) The user is redirected to the itsme® website. The query includes the `scope` of "claims", the list of private data fields of the user the authentication service requires.
b) The user enters her phone number on the itsme® sign-in page.
c) The user is asked to authenticate using the itsme® app on her mobile device.

If the connection is initiated from a **mobile device**:

a-c) the user is directly redirected to the itsme® app. This redirection also includes a query with the required `scope`.

Then, on the mobile app itsme® :

d) The user consents to the release of personal information included in the `scope`.
e) The user signs by providing her credentials (itsme® code, fingerprint or FaceID).

In case of an error, a message is displayed to the user. Else, after successful authentication :

f) an authorization `code` of 36 characters is provided in the response of the query to the AS with a lifetime of 3 minutes.
g) the AS's back-end will interact with the itsme® OIDC (OpenID Connect) endpoint to obtain user information.
    i) AS sends a POST request using the authorization `code` obtained from the initial user response. The itsme® API is designed to respond only to authorized applications, so the request must be signed by the AS.
    ii) The response to this request includes an `id_token` encrypted using the public key of the AS, along with an `access token`.
    iii) The authentication service decrypts the `id_token` to retrieve the personal information specified in the initial `scope` of the request.
    iv) Later, the `access token` can be used to re-query the user's consented data throughout the duration of the token's validity, which typically lasts for about one hour. However, this should not be necessary for this authentication protocol

    Given that the authentication service likely already has access to personal information, the request `scope` for data may be minimal. Retrieving only the national number might suffice to access voter information in the citizens' database. Nonetheless, it could be beneficial to also request basic information like full name, birthdate, and address for verification against the eID card data.
h) The AS queries its database to check whether the user is an eligible voter and has not received her certificate yet. It also extracts the locality of the voter, to provide the correct list of candidates and create an appropriate certificate.

i) If eligible, the service will request a certificate from the Certificate Authority *CA_Electors*, which will be used for interactions on Fabric's network. This certificate includes an OU (Organizational Unit) field with the municipality code, enabling voting in the corresponding chaincode.
j) Once the user certificate has been generated, the database is updated with this certificate and a value indicating that the voter has received its certificate. Therefore, an eligible voter cannot request multiple certificates, to preserve unreusability.
k) If there was no error in the process, the user is redirected to the voting website or app, and a success message is displayed. The certificate is securely stored on the device, along with the locality code.
l) In the event a voter loses their certificate (e.g. clearing the cookies, or for any technical issue), she can request a new one. The *CA_Elector* will revoke the previous certificate and regenerate a new one.

6.4 Cryptographic protocol

6.4.1 Design motivations

We decided to implement the protocol originally proposed by He and Su in *A new practical secure e-voting scheme* [73] from 1998, which was later adapted for distributed ledger technology by Kirillov *et al.* in [86]. While this choice may appear unexpected in light of Figure 5.5 from the conclusion of our literature review in Section 5.3, there were specific motivations behind it. Neziri's model, despite ranking highest in our comparison, lacked detailed technical insights for implementation and did not provide the information required to assess its scalability. Agora, the second-highest-ranking solution, operates with proprietary software as a commercial entity, limiting our ability to freely adapt their model without purchasing the complete solution. The model proposed by Hjálmarsson did not respect multiple requirements, such as coercion resistance, and tally period, and had serious limitations regarding anonymity. Khan was less satisfactory across various requirements. Mukherjee's model, another potential candidate, aligns closely with several requirements but faces significant scalability issues and involves transaction fees. These problems could be solved by adapting using a permissioned blockchain. However, its lack of detail on the Zero-Knowledge Proof (ZKP) mechanism was a limiting factor. Ultimately, the decision was made to adopt Kirillov's model due to its compatibility with both on-site and online voting formats and its overall satisfactory performance in various key requirements. However, it's important to note that this model falls short in providing receipt-freeness. Additionally, a notable limitation is the requirement for users to submit data at various times for vote validation but we will explain later that we can make small adaptations to solve this issue.

6.4.2 He and Su e-voting protocol

Before diving into our adaption with a model adapted for Belgian elections, let's first define the original protocol defined by He and Su [73]. This protocol is composed of different steps:

Registration (1/2) get blind signature: The authority has established a list of eligible voters before the election. To register, voter V generates a pair of keys (D_V the private "voting" key and E_V to public "tallying" key) and a random number R. Then she computes $E_a(R) * (h(E_v))$ with E_a the authority's public key and sends it to the authority A while authenticates with its real identity.

Authority A verifies that V is an eligible voter. If this is the case, A blindly signs the data with its private key $D_a(E_a(R) * (h(E_v)) = R * D_a(h(E_v))$ and sends it to V. Blind signatures are defined in Section 3.1.3.

V removes R by multiplying the response by $1/R$, verifies the signature of A with $E_a(D_a(h(E_v))) \stackrel{?}{=} h(E_v)$, then $D_a(h(E_v))$ is now a valid signature on her tallying key E_v.

Registration (2/2) register public tallying key : Voter V sends anonymously to the tallier T her public tallying key and the associate blind signature $\{E_v, D_a(h(E_v))\}$. The tallier T checks the authenticity of the key E_v by verifying the signature $E_a(D_a(h(E_v))) \stackrel{?}{=} h(E_v)$. If it is correct, T stores E_v as an authorized key, and at the end of the registration phase, publishes all the authorized keys. It is worth noting that, if $\{E_v, D_a(h(E_v))\}$ was transmitted in an anonymous way, then the identity of the voter is preserved. If R is kept secret, the authority cannot recompute the data $E_a(R) * (h(E_v))$ that it signed.

Voting: Voter V encrypts its ballot B_v with a symmetric "voting" key K_v, signs it, and sends anonymously along with her public key registered in the previous step to T $\{E_v, K_v(B_v), D_v(h(K_v(B_v)))\}$. The tallier T verifies the signature $E_v(D_v(h(K_v(B_v)))) \stackrel{?}{=} h(K_v(B_v))$, checks that the public key was indeed registered, and publishes the received data in the positive case. All voters can individually verify that their vote is correctly published. If authorized in the election policy, a voter could modify her vote, and this new vote would replace the previous one sent with the same E_v.

Tally: At the end of the voting period, all voters send to T their voting key K_v and sign them $\{E_v, K_v, D_v(h(K_v))\}$. T verifies the authenticity of K_v with $E_v(D_v(h(K_v))) \stackrel{?}{=} h(K_v)$, then decrypts $K_v(B_v)$ with $K_v^{-1}(K_v(B_v)) = B_v$ and publishes all the authenticated data $\{B_v, K_v(B_v), K_v, D_v(h(K_v(B_v))), D_v(h(K_v)), E_v\}$ for universal verification. Anyone has then the possibility to

6.4.3 Adaptating for distributed ledger

We will now adapt this protocol to our model with a distributed ledger, and leverage the possibilities offered by Hyperledger Fabric. As mentioned, this solution is inspired by the work of Kirillov *et al.* in [86]. At the start of this protocol, we consider that election administrators have already set up the network (Section 6.2), including creating and configuring the channels and chaincodes, that authorized organizations have generated their certificates, are running their peers, and joined their respective channels and that an eligible voter has authenticated with the authentication service and obtained an X.509 certificate issued by $CA\_Electors$ (Section 6.3).

Registration (1/2) get blind signature: The voter has already been identified, and can therefore directly request a blind signature. The voter client generates a pair of public (E_v) and private (D_v) RSA keys (see Section 3.1.2) and a random number R, as described in He and Su protocol. She computes $E_{cc,i}(R) * h(E_v)$ with $E_{cc,i}$ the public key of the chaincode i of the corresponding municipality. The client creates a transaction proposal, interacting with the chaincode CC_i corresponding to the appropriate municipality, and sends it to the peers endorsing the chaincode, as defined in the channel policy. These peers verify the certificate (that it is correctly formed and not tempered, signed by the Root Certificate Authority $CA\_Electors$ or a trusted intermediary CA, and that it has not been revoked), and execute the transaction in the chaincode, which checks that the OU (Organization Unit) included in the certificate corresponds to this chaincode CC_i. If it is correct, the chaincode blindly signs (see Section 3.1.3) the data with its private key stored in the private data collection (a type of data stored only on the authorized nodes, used to endorse transactions but not transmitted to the ordering nodes).

$$D_{cc,i}(E_{cc,i}(R) * h(E_v)) = R * D_{cc,i}(h(E_v))$$

The ledger is then updated with a record that the voter has received her blind signature. As above, the voter removes R by multiplying the response by R_{inv} the inverse modulo of the $E_{cc,i}$, verifies the received signature with $E_{cc,i}(D_{cc,i}(h(E_v))) \stackrel{?}{=} h(E_v)$ and $D_{cc,i}(h(E_v))$ is now the voter public key's blind signature. Note that this step could also have been executed off-chain. In this case, after authentication to the AS, the

user could send the data and the AS would directly provide the blind signature. With this method, the blind signature is provided directly at the same time as as the certificate. It simplifies the process as only one database stores wich voters have received the certificate and the blind signature, instead of one managed by the AS and one on the distributed ledger.

Registration (2/2) register public key: The voter sends her public key E_v to CC_i along with the received blind signature anonymously using Identity Mixer (see Section 3.1.7). Idemix is a cryptographic protocol that allows users to confirm their membership in a group or their credentials without revealing their full identity or any extra information using Zero-Knowledge Proofs (ZKPs). An explanation of how Idemix is used in our network is provided just below.

CC verifies the signature simply with $E_a(D_a(h(E_v))) \stackrel{?}{=} h(E_v)$ and stores the voter's public key. All the registered public keys are publicly accessible, so any voter can verify that her public keys have been registered.

Public key revocation: The voter has the possibility to revoke a registered public key. This might be the case if the signature was lost because of a technical problem for instance, if the voter changes its mind and prefers to vote on-site, or if she wants to modify her vote. This phase is not anonymous, so, as explained later, modification of a vote can only occur if the private key is not yet transmitted. The voter must simply send her public key E_v to revoke it while authenticating her identity. The CC will mark E_v as revoked, and the user will recover her right to a new blind signature. At this moment, the revoked voter can either vote on-site or regenerate a new pair of keys and return to the first registration step.

Voting: Once the voter has registered her public key, she can send an encrypted vote to CC_i. Once again, this transmission is done anonymously using idemix. In the He and Su model, the encryption of the vote is done using a voter's symmetric tallying key. In its adaptation, Kirillov directly encrypts the ballot with the voter public key. At the end of the day, this does not change the tally and the results. For the sake of simplicity, we will use the Kirillov method and consider that the vote is encrypted with the voter's public key. So the voter anonymously sends the registered public key E_v, the ballot padded with a random string $R0$ encrypted with the key E_v and signs it : $\{E_v, E_v(B_v|R0), D_v(h(E_v(B_v|R0)))\}$. The chaincode (now playing the role of the tallier in He and Su protocol) verifies the signature $E_v(D_v(h(E_v(B_v|R0)))) \stackrel{?}{=} h(E_v(B_v|R0))$, checks that E_v was a registered public key, and then stores the received data.
The voter will then send her private voting key between the moment she has sent her encrypted vote and the end of the election period. Further details regarding this process are addressed in the additional remarks of the following subsection.

Tally: At the end of the election period, all the private keys D_v are transmitted to the CC_is and published for universal verification. Each voter can verify that her private key has been received and her vote can be decrypted. CC_i decrypts all the votes $D_v(E_v(B_v|R0)) = B_v|R0$, remove the random string and counts the results. The results of each chaincode are then aggregated and the final results are published.

6.4.4 Additional remarks on this protocol

Identity Mixer in Hyperledger Fabric

The concept of Identity Mixer (idemix) was introduced in Section 3.1.7. Idemix plays a central role in our proposed solution of a voting protocol. They are used to prove that a voter (the user) is authenticated by a CA (the issuer) and indeed has a voting right from a municipality (the disclosed attribute) as required by the MSP (the verifier) without displaying its identity (Common Name or public key) nor any extra unnecessary information (contained in the X.509 certificate) to protect its anonymity.

Luckily, Idemix tools are already built-in in Fabric. The Fabric Java SDK allows a client to request idemix credentials from a single API call, and the Fabric CA servers can be configured to automatically generate these credentials. The implementation of Idemix in Fabric uses a pairing-based signature scheme that was proposed by Camenisch and Lysyanskaya in [25].

Transactions sent via Idemix in Fabric are untraceable, meaning that the identity of the transactor is undetectable. As a result, even if the same user were to send multiple transactions, their actions would remain unlinked and indistinguishable from one another. This characteristic of non-traceability is not an issue within the voting protocol since it's designed to ensure that an eligible voter can register only one voting public key, thus preventing multiple voting instances by the same individual (unreusability). Idemix credentials always include the `role` attribute (admin or member) and the `OU` (Organizational Unit) attribute. These ensure that users do not exceed the access rights initially granted to them. Even when transactions are initiated using Idemix credentials, a chaincode can verify the user's organizational affiliation through the `ou` attribute. Our approach leverages this feature, allowing anonymous users with Idemix credentials to interact exclusively with chaincodes linked to their `ou`. In other words, an anonymous voter is still associated with its municipality of residence, and can only interact with the chaincode of its municipality.

Vote modification and transmission of the private keys

The possibility of letting voters modify their vote is optional and is left to the choice of the election authority, based on local regulations. Authorizing vote modification could potentially improve the acceptance of the online vote, but forbidding it would make the management of the voting process easier, and prevent the non-transmission of private keys.
If vote modification is allowed, just after she has sent the vote, the voter now has the choice between either immediately transmitting the private key for ballot decryption or waiting in the case she would modify her vote later. In the second case, the voter can later revoke her public key, and go back to the first step of registration by generating and registering new keys. As the key revocation is not anonymous and requires revealing the identity, a vote cannot be modified once the private key has been transmitted. Otherwise, the decrypted vote would be linked to the voter. If not done immediately, the voter must transmit her private key by the end of the election. A deadline must be legally defined, behind which private keys cannot be accepted anymore. If some voters did not transmit their private key in time, their vote would be considered invalid, as they could not be decrypted.
If vote modification is not allowed, the voter would directly transmit the private key after casting the vote.

If a voter decides or automatically have to directly transmit her private key, it is securely transmitted to a server maintained by the election authority for safekeeping until the election concludes. An alternative method involves storing the private keys in a private data collection of HLF, ensuring they are securely held within authorized nodes, yet inaccessible from ordering and reading nodes. In the meantime, these keys are not made public, thereby preserving the confidentiality of the election results until the voting period is over. The server can decrypt the votes locally but is unable of decrypting the voter's identity (as the transmission is done anonymously). At the end, this server publishes all the decryption keys. As a consequence, the authority has the capability to decrypt votes locally before the voting period concludes. However, as long as the preliminary results are kept confidential, this parallels the current system where polling stations have access to early results. As in today's system, should a leak occur, only these preliminary results are revealed.

If the early decryption of the votes locally by the election is not a desired property, an alternative approach involves encrypting the private voting keys with a shared key held by a group of trustees. At the end of the election, a designated threshold number of these trustees would gather to collectively decrypt all the voting decryption keys.

Different types of public and private keys

It is important to note that the user has distinct types of asymmetric public and private keys and to understand their respective roles to avoid any confusion. The pair of public (E_v) and private (D_v) keys described in the protocol are only used for encrypting and decrypting the voting ballot. As multiple elections are typically held on the same day, one pair of keys is generated for each election. It must not be confused with the public key of a client or a peer interacting with the HL Fabric network. This public key is indicated in the X.509 certificate signed by a CA authorized on the channel policy. The user keeps the associated private key secret and never shares it! These keys are also distinct from the asymmetric keys used during the TLS handshake to establish an HTTPS connection between two devices over the internet.

Do the 3 steps consecutively

An issue with Kirillov's model was that the process was done in three phases (registration, voting, and submission of the private keys). If the voter did not complete the three steps, it resulted in absenteeism (registration but no voting) or an invalid vote (voting but non-transmission of the private key). This issue can be solved by consecutively executing the three steps: requesting a blind signature, registering the public key, and casting a vote, with a short delay of a few seconds between each to ensure the transactions are not linked to the same user. For systems not permitting vote modification, the private key could be transmitted immediately after voting, using the previously described method. This approach simplifies the process for voters and eliminates the risk of failing to submit private keys

Synchronizing the list of eligible voters

As mentioned in Section 6.3, a database must contain the list of eligible voters, and indicate whether they have already received their certificate. This database is essential to prevent dual voting both online and on-site. Polling stations require real-time access to this database to verify if an eligible voter retains the right to vote in person and to update the voter's status immediately after they have presented themselves on-site. This system must be designed to prevent simultaneous authentication from a voter's smartphone and in-person voting. One security measure could involve synchronizing the database to the exact second, effectively disallowing concurrent voting activities.

An alternative approach might involve requiring voters to request their certificates and blind signatures several days before the election, then removing them from the list of eligible voters. However, this strategy would compromise the convenience of completing all voting steps in a single, continuous process, as discussed just above.

If implementing such a real-time database system in all voting stations simultaneously proves challenging, a phased approach could be adopted. Initially, online voting could be introduced only in constituencies equipped to integrate this database with their polling stations. This method would allow a controlled, manageable transition, starting with pilot programs in select municipalities and gradually expanding. Such a transition ensures there is no obligation to launch online voting nationwide at once, providing time to refine the system based on initial experiences.

6.5 The voting process for the voter

Here is a summary of the main steps of the election process from the user's perspective.

Before the Election:

1. **Registration**: The user must register on the electoral list if necessary.
2. **Account Setup**: The user sets up an itsme® account.

On Election Day:

3. **Connection:** The user connects to the website of the online election.
4. **Authentication**: The user authenticates using itsme®. Behind the scenes, the Authentication Service (AS) retrieves the personal data from the itsme® server and creates a certificate. Both this step and the previous can be done either on election day or in advance to reduce workload.
5 **Candidate Selection**: The user retrieves the list of candidates according to their constituency and selects their preferred party and the list of candidates within this party.
6. **Key Generation and registration**: In the background, the client software creates pairs of keys for each election, requests a blind signature, and registers the public key for the first election.
7. **Vote Encryption and Transmission**: Once the choice is made, the client concatenates the party with the candidates, adds a random string to prevent forgery, encrypts the vote, signs it, and transmits it anonymously to the ballot.
8. **Vote Verification**: To verify that the vote was cast as intended, the voter can use a QR code. This QR code, containing their vote, the public key and the random string, can be scanned with a smartphone. The client on this second device can then replicate the encryption process and verify if the encrypted vote matches the expected result.
9. **Private Key Transmission**: The private key for decrypting the ballot is sent.
10. **Repetition for Multiple Elections**: This process is repeated for each election happening on the same day.

At any time:

11. **Election verification:** The voter can verify that her votes were well registered and counted via an external blockchain explorer provided by reader nodes.

This streamlined process is user-friendly, with the technical aspects occurring seamlessly in the background. The time taken for candidate selection typically allows for any required cryptographic computations (like ZKP) or mixnet processes to be completed without causing delay or inconvenience to the user. The user doesn't need to have any technical knowledge in computer science, blockchain, or cryptography to be able to vote online.

6.6 Executive Summary

In this chapter, we have proposed a model of an online voting system adapted for Belgian elections, based on distributed ledger technology, more specifically the Hyperledger Fabric blockchain framework. We started by motivating the design choice regarding the use of a permissioned blockchain and the Fabric framework (Section 6.1). We then proposed a complete network architecture of the Fabric network (Section 6.2), including organizations and their Certificate Authority structuring, defining the type of institutions taking part of the network, the type of nodes each of them can run, and the permissions and channel policies defined in the MSPs. Each municipality is associated to a distinct chaincode. In Section 6.3, we described the process of how a voter authenticates to the authority using itsme® and receives a certificate to prove her right to vote to her municipality's chaincode. The protocol used to vote anonymously and tally the result is detailed in Section 6.4. The voting process from the user's point of view is detailed in Section 6.5.

Chapter 7

Solution analysis, discussion and future works

7.1 Analysis of the solution

Our analysis will assess how our proposed solution aligns with the requirements outlined in Section 4.1. We'll follow the same structured approach, analyzing first the general election requirements of an election, then the e-voting and i-voting specific requirements, and finally the requirements specific to Belgian elections.

7.1.1 General election requirements

Integrity : Integrity of the votes is guaranteed by the inherent tamper-proof nature of blockchain technology. Once a transaction, such as a vote, is incorporated into a block by the current leader in the Byzantine Fault Tolerance (BFT) consensus protocol, and then disseminated to other ordering nodes and all committing nodes, it becomes immutable. Any attempt to alter or delete a vote would lead to a chain inconsistency, detectable when compared with the agreed-upon chain held by other network nodes. The permissioned nature of the network, managed exclusively by authorized peers from meticulously selected organizations, further protects against the likelihood of a widespread network compromise.

Eligibility: Only voters with a certificate issued by the $CA_Electors$ or a trusted intermediate CA can interact with the network and cast a vote. This certificate is issued after a successful authentication using itsme®. It is the responsibility of the election authority to confirm the eligibility of the authenticated voter against official voter lists. Moreover, independent auditors are tasked with ensuring that certificates are not fraudulently issued to non-existent or ineligible voters.

One-human, one-vote (*un-reusability*): Once an eligible voter has received an X.509 certificate, it can only request one blind signature, therefore registering only one public key and ultimately casting only one vote per election. In other words, the model mathematically ensures un-reusability of a certificate, but the issuing of these certificates is the Certificate Authority's responsibility.

Anonymity and Privacy (*un-traceability*): Each ballot is associated with a unique voting public key, which is registered through a process that ensures anonymity, utilizing a combination of blind signatures and Identity Mixer. The blind signature prevents any direct link between the authorization to register a voting public key and the identity of the voter who requests this signature. On the other hand, Identity Mixer prevents the possibility of tracing back the voting public or private key used in the protocol to the individual voter.

A limitation of this solution, considering the extreme importance of guaranteeing anonymity in the context of online voting, is that if the Certificate Authority issuing the idemix credentials also manages the chaincode, runs peers, and stores network logs,

there is a potential risk that a link between the IP address requesting the certificate and the one sending the transaction could be established. A solution to this problem could be the use of a VPN. However, the user should change the route at every idemix request, which is not always possible when successive operations are done directly one after the other, and the technical knowledge and capabilities are inaccessible for most of the population, especially for non-tech-savvy people. Another solution would be the systematic use of Network Mixers (mixnets) (see Section 3.1.6) for every transaction in the Fabric network. This last solution would permit to hide the source and destination of every transaction and add an extra layer of privacy but at the cost of additional complexity in the network management. Finally, a third solution would be to clearly separate the roles played by election authorities into distinct institutions and establish strict procedures to prevent cross-analysis of IP addresses

Freeness : The proposed online voting process incurs no cost for the voters. In today's digital age, most individuals have access to the internet either through a fixed monthly subscription, which does not charge based on data usage, or through widely available free Wi-Fi hotspots. This accessibility ensures that internet connection costs do not pose a barrier to participating in the voting process. However, for those who may not have internet access, the option to vote on-site is available as an alternative. Moreover, the use of a permissioned blockchain eliminates the concern of transaction fees typically associated with public blockchains, as this system does not incorporate a native cryptocurrency.

Auditability: The system's auditability is ensured by auditors operating Reader nodes, which maintain copies of the blockchain. This setup enables thorough verification of the voting process's integrity. Additionally, the auditing scope extends to the authentication service, the certificate issuance process, and any other components involved in the voting system. Also, all the source codes (client app, smart contract, ...) should be open source and independently audited. Note that the Hyperledger framework is itself open source.

7.1.2 e-voting and i-voting specific requirements

Verifiability: Our proposed system is designed to be end-to-end verifiable, ensuring voters can confirm their votes are accurately recorded and counted. Firstly, a voter can verify that their vote has been cast-as-intended, by scanning for instance a QR code containing their vote, their public key, and the random string with a smartphone, replicating the encryption process, and confirming that the result matches the encrypted vote. Then, each voter has the ability to confirm that their vote is correctly recorded-as-cast on the chaincode, and that their private key was correctly recorded too. Lastly, the system allows for universal verification where any observer can validate that all ballots are accurately tallied-as-registered following the decryption process. Reader nodes ensure complete transparency by enabling public access to all data, allowing any individual to view, analyze, and verify the data regardless of whether they are authenticated on the Hyperledger Fabric network, or not.

Receipt-freeness: Unfortunately, this requirement is not totally respected. Indeed, as votes are stored along with public keys, a voter can easily view the candidate she voted for by querying the blockchain to get the vote associated with her public key. Verifying this simply the content of their vote might be good for most people, as it is easier to verify the content of a decrypted votes linked to its own public key on a blockchain explorer than trusting or reproducing complex cryptographic operations. However, this is at risk of vote coercion or vote buying as one could be tempted to vote for some candidate for money and prove her vote. This balance between verification and non-coercion was already discussed in Section 4.1.4. But how to prove that a public key actually belong to a voter ? A voter could demonstrate ownership of a public key by signing a random piece of data with their private key, and others could verify this signature against the public key. However, this process might be too complex for

the average person, making widespread coercion through this method unlikely. A more straightforward method for a sophisticated coercer would be to directly demand the voter's private key.

Another risk with this "non-receipt-freeness" is that if a voter's private key is compromised (either by human error or hacking), it is possible to link the private key to its corresponding public key, despite RSA's inability to directly derive a public key from a private key. Since each municipality's chaincode contains only a limited number of public keys, a brute-force approach could be employed. Indeed, the number of private keys for one chaincode is at most equal to the number of eligible voters in this municipality. By signing a random string with the compromised private key and then verifying this signature against all public keys registered in the chaincode, the corresponding public key can be identified. Remind that we created one chaincode per municipality as it is the smallest unit of election constituency, but it could be larger for other types of elections (like House of the Representatives) and in that case, a public key would be a bit more difficult to brute-force (with thousands of RSA signature per seconds, it still shouldn't exceed a few minutes).

Availability: The chaincode policy dictates the beginning and conclusion of the voting period. Although voters could potentially authenticate, request a blind signature, and register a public key before the voting period, actual vote casting is restricted to the designated timeframe. The submission of private keys for decrypting the votes must occur prior to the voting period's deadline. Also, the network's decentralized structure significantly enhances its robustness. This approach contrasts with a centralized server system, which, as a singular point of failure, poses a greater risk of causing system unavailability.

Hybrid Voting Approach: Voters have the flexibility to choose between online voting through our system or traditional on-site voting. They remain eligible for on-site voting until they request an online voting certificate. If a voter initially chooses online voting but decides later to vote on-site, she can revoke her online certificate and public key, as long as her private key hasn't been sent.

Accessibility: Our hybrid voting model boosts overall accessibility by allowing those unable to vote in person (due to reasons like living abroad, work commitments, or mobility issues) to vote easily online, eliminating the need for a proxy. Additionally, it retains traditional voting methods for those less familiar with technology or lacking internet access. The voting process was adapted to be as quick and easy as possible, and the user interface of the voting mobile app or website must be user-friendly.

7.1.3 Belgian elections-specifics requirements

Multiple elections on the same day: As discussed in the voting process from the user point of view, the system is thought to be used when multiple elections happen on the same day. The user only has to authenticate once. The cryptographic protocol to generate and register keys is hidden in the background and do not require human intervention. All the voter has to do is select the candidates for each election.

Note that it is theoretically possible from the model perspective for a voter to choose to vote online for one election (say, the federal elections) and then revoke their keys for other simultaneous elections (like the European Parliament and Regional elections) to vote in-person. We can doubt the purpose of doing so and don't see any valuable reason to do so, but it would be theoretically possible. Current voting station procedures do not support partial on-site voting for selected elections. Thus, any implementation of this feature would require adjustments to the on-site voting process, rather than the online system.

Mandatory vote: It is possible to get the list of electors that have voted on-site, and the list of voters that have requested a certificate and a blind signature. There is a limitation in this system, as it is not possible to know whether a voter having requested

a blind signature has indeed cast a vote and transmitted her private keys. However, one might consider that is a tolerated limitation, considering that the absentees are usually not prosecuted. These non-votes could be considered either as absenteeism (no vote cast), or invalid (decrypting private key not sent).

Blank votes: Blank votes are totally possible with the system. When the voter decides not to select any party, an option "blank vote" is proposed. If selected, the ballot will contain a `null` value padded with the random string. It will then be encrypted and sent indistinguishable from a normal vote. During the tally phase, after decrypting the votes, the system will count blank votes. This complies with the law which recognizes blank votes, although they don't influence the allocation of seats.

Preference votes: In Belgium, preference voting offers a unique aspect compared to other major countries' voting systems. Our system fully supports this. When a voter casts their vote, it's essentially a structured string comprising the chosen party ID, followed by IDs of the selected candidates, and a random string for added security.

7.1.4 Optionnal features

Flexibility : The model is suitable for all types of elections currently held in Belgium such as federal and municipal elections. But it is flexible enough to be adapted to any type of votes (like a hypothetical referendum or public consultation). As votes are represented as structured strings they can store any type of data, whether they involve party and candidate IDs, or a list of propositions, or use weighted voting methods like Borda or Condorcet.

Modification of a vote: As mentioned in section 6.4.4, voters could be given the option to modify their vote until they send their private keys. However, this practice is not advised. Allowing vote modifications raises the risk of voters forgetting to submit their keys before the election concludes, which would inevitably result in their votes being invalid.

7.2 Security analysis

A non-exhaustive list of the security risks related to blockchain voting was presented in Section 4.3.3. This section evaluates how the proposed model addresses these concerns.

Risks inherent to online voting

When voting is conducted through personal devices such as computers or smartphones, the security of these devices cannot be guaranteed. A crucial question arises: what if a voter's device is compromised? Thanks to the end-to-end verifiability of our proposed system, even if malicious malware attempts to cast a fraudulent vote by misusing the authentication certificate or the voting private key, the voter retains the capability to confirm that their vote has been accurately cast and recorded. If a difference is found between the vote and the voter's intention, this issue will be immediately detected. This underscores the importance of encouraging voters to verify their votes, even if they trust the system.

Phishing attacks, where voters are fraudulently directed to a fake website that copies the appearance of the official voting site, are often cited as a threat in online voting. These fake sites aim to either steal user credentials or discover voting preferences. In our system, as the authentication is done using itsme®, and as only authenticated services can send requests to the itsme® OIDC endpoint, a fraudulent website couldn't steal the voter's credentials, nor request a certificate. Nonetheless, the potential for discovering a voter's preferences remains a concern. It is therefore important to educate the people to only access the official URL, and not click on suspicious links.

Another risk inherent to online voting is the risk of coercion from home. While the vote is normally done confidentially in the secret booth, there is no way to be sure that a voter can vote secretly at home, without pressure or influence for the voter's relatives.

Regarding availability attacks like DDoS, a decentralized network may offer enhanced resilience due to its distributed nature, allowing multiple nodes to handle incoming transactions.

However, our model does not mitigate risks associated with widespread attacks targeting a nation's entire internet infrastructure. Such large-scale threats are inherent to any online voting system, and the implementation of blockchain technology does not offer a solution in this regard.

Risks specifics to blockchain technology

A 51% attack is unlikely in our context, as such attacks typically affect permissionless blockchains. In these systems, any participant can join and potentially accumulate over half of the computational power (in Proof of Work) or stake (in Proof of Stake). In contrast, our permissioned network restricts entry to entities authenticated by an authorized Certificate Authority (CA) or those explicitly authorized in the channel policy, making such attacks infeasible.

However, a comparable threat in our scenario involves gaining control over the ordering service. We utilize Fabric's BFT ordering protocol, based on SmartBFT, a variant of the Practical Byzantine Fault Tolerant (PBFT) consensus protocol. This protocol can work as long as two-thirds (plus one) of the nodes are honest. This protocol remains functional as long as more than two-thirds of the nodes are honest. If an adversary were to control exactly one-third or more of the total nodes, denoted as N, they could disrupt the consensus process. Specifically, they could prevent reaching the required $2N/3$ vote threshold for proposals, effectively obstructing consensus. As a consequence, the adversary could manipulate the inclusion of new transactions in blocks, either halting the service entirely or selectively blocking certain transactions. It is important to note that even if an opponent controls the network, they cannot generate transactions on behalf of others, as they cannot replicate the necessary signatures without access to the private keys.

In our proposed model, the risk of a Sybil attack, where an individual creates numerous fake identities within a network to gain influence, is mitigated by the fact that identities are strictly regulated, allowing only authenticated users with certificates issued by $CA\_Electors$ or a verified intermediate CA. However, this is akin to the concerns addressed under the *Eligibility* requirements, pointing out the election authority's responsibility to issue certificates exclusively to eligible voters. A scenario where the authentication service is compromised could lead to an influx of fake voters, resembling a Sybil attack.

Wallet-based attacks, in this context, refer to the same issue as explained above, where the client's private key could be stolen by malware on an infected machine, via social engineering, or simply by other people having physical access to the unattended device.

A smart contract attack, in this context, could be for instance an attacker discovering a vulnerability in the chaincode, to manipulate the decryption phase or alter vote counts. However, since all transactions are recorded, it could be feasible to retrospectively recompute the correct results after fixing the issue. A more significant threat would be an exploit allowing the anonymous registration of additional public keys or casting unauthorized voting. To mitigate this, an in-depth technical audit of the smart contract is imperative prior to the election, ensuring the chaincode is safeguarded against potential vulnerabilities.

The paper [106] argued that an issue regarding the security of blockchain was that decentralization induces a slow reaction, and therefore a long reaction time to mitigate any type of security problem. While this may be a concern for permissionless decentralized blockchains, our model offers a more agile response mechanism. In the event of a security issue, such as infected and dishonest acting peers within an organization,

a coordinated response among partner organizations is required. One possible course of action could involve revoking the authentication of the affected peers. This could be achieved by modifying the channel MSP to exclude the problematic organization from the network. The threshold for the number of organizations needed to amend the MSP is predetermined in the channel policy. This threshold should be set high enough to prevent a single compromised entity from unilaterally altering the policy, yet low enough to allow for quick action in an emergency. Predefining procedures for such scenarios is critical to ensure prompt and effective responses, thereby enhancing the overall robustness of the system.

7.3 Discussion

Cost estimation

Money is often the lifeblood of a project. If a revolutionary idea is too expensive, it won't be able to develop. We cannot provide a precise cost that covers every aspect of the project, but we can offer some insights for a general estimation. To begin, let's consider the cost of a conventional election. The expense of conducting an election in Belgium is currently approximately 10 million euros. Given that there are around 8 million eligible voters, this equates to slightly over 1 euro per voter. The cost encompasses the setup and dismantling of polling stations, remuneration for assessors and presidents of polling stations, paper ballots, electronic voting machines, and additional electronic equipment such as e-Urns, magnetic cards, and computers for each station. These machines are re-used from year to year but have to be maintained, and sometimes replaced.

In [88], the authors analyze the costs of different ways of voting. They concluded that in Estonia, the cheapest votes are those cast online. These results correspond with the study of R. Krimmer *et al.* [88] who concluded that an internet-based voting system was the most cost-efficient voting system.

The cost of an online election can be divided into two origins: the development of the system, and the cost of hosting an election. For our model, the cost for the development includes the research to provide a secure way to vote to respect the requirements cited in this document, the development of the software (client voting website and app, authentication service, and other apps for monitoring and managing the election), and the smart contract. Then there must be an audit reviewing and analyzing the securing of those, and extensive testing. Creating a database, from which can interact the authentication service but also the voting station, establishing the list of partner organizations, etc. According to [44], the development of the different online voting systems implemented in the world ranged between half a million and ten million. Once this solution is developed, the cost for each election is much smaller: Paying a dozen partner organizations to run some peers during the election day, a team of cloud engineers monitoring that everything goes as expected, and auditors. A few servers are necessary for the authentication service, redirecting some requests and transactions, and running some nodes in the networks. The voters vote from their own devices. In total, these costs are really low per election. The greater part of the cost would probably be part still on-site to manage the voting station. At first, there is a risk of duplication of the cost: almost the normal price of an election to host the voting station, in addition to the new cost of this online solution, plus the initial development costs. But if the system succeeds and this solution attracts more and more voters to vote online, the number of voting stations could decrease, reducing the costs, and, in the long term, online elections could become cheaper once the development costs have been amortized.

Legal considerations

The laws governing elections in Belgium define many aspects of the current system, with elections that can be held both on paper or on electronic machines with voter-verifiable paper audit trail (VVPAT). However, the current legal framework does not

accommodate online elections, necessitating significant legal revisions to integrate this method. While this paper does not provide a detailed list of specific legal articles for amendment - a task best suited for legal experts - but this section aims to open the discussion on some legal considerations to have.

In the event that online voting is adopted, it will be essential to enact laws specifically tailored to support this new system. This includes defining the legal framework for online voting, setting standards for security, privacy, and data protection, and voter verification and authentication. It is also necessary to establish clear procedures to delineate roles and responsibilities among various stakeholders involved in what this paper refers to as the "election authority." It's crucial to create guardrails to prevent the concentration of power in the hands of single individuals or entities, and to prevent the cross-analysis of IP addresses, which could potentially compromise the anonymity guaranteed by Idemix in the voting protocol.

Regarding security, legislation should establish stricts and robust security protocols for the online voting platform. It's crucial to define a clear process and criteria for establishing the list of trusted partners. Additionally, legal provisions need to be in place to address system failures, security breaches, and other unforeseen incidents during the voting process. This is particularly important in a slightly decentralized system where actors must cooperate together to update the protocol and policies. Furthermore, the laws should specify penalties and legal repercussions for electoral fraud and various online threats, including hacking, phishing, and impersonation.

Regarding transparency, legal regulations should establish specific criteria for authorization to operate "reader nodes." These laws should also determine the scope of data that must be made publicly accessible, including guidelines on how this data can be accessed. The legal framework may require that the voting software be predominantly open-source, allowing exceptions for certain security-related components to remain closed-source. Additionally, audit procedures need to be defined, including software audits, real-time monitoring of elections, and in-depth post-election reviews.

Regarding privacy and data protection, existing laws may require updates to specifically address voter data management in an online voting scenario, ensuring compliance with GDPR.

For accessibility, legislation should ensure that online voting platforms are inclusive, catering to all voters, including those with disabilities. This may involve legal requirements for user-friendly interfaces and support for assistive technologies, along with voter education campaigns about the new system and its security.

On authentication, concerns arise over the reliability of authentication provided by the itsme app. Although itsme®'s signature is considered a Qualified Electronic Signature (QES), equivalent in value to a handwritten signature, there are questions about whether this level of authentication is adequate for granting voting rights. The risk of impersonation and potential vote theft is a significant concern in online voting, particularly if an individual gains access to another person's voting devices and credentials. These concerns underline the importance of defining penalties against this type of fraud and educating voters on security measures.

Also, the use of online voting might change some aspects of current regulations. For instance, the current closing times for voting stations are set at 2 pm for paper voting and 4 pm for electronic voting, primarily due to the absence of manual counting in the latter. With online voting, these closing times might need reevaluation. Could the online voting period be extended beyond 4 pm, considering the elimination of the need for physical dismantling of voting stations? Another aspect to consider is the handling of ballots post-election. Traditionally, ballots are destroyed after a legal retention period if no appeals are made, and the final results are then officialized. In the proposed online model, anyone could theoretically retain a copy of all ballots, indicating a shift in how election results are archived and verified.

Also, should it be tolerated that voter could produce a "receipt" of their vote? In traditional voting, voters place their ballots in an urn without keeping any trace of

their vote. While photographing one's ballot inside the voting booth is technically prohibited in Belgium, it's challenging to enforce this rule due to the secrecy of the voting booth. Our proposed model is not *receipt-free*, and voters could therefore prove their vote. But more generally, online voting might inherently allow voters to retain some form of digital "receipt" of their vote, such as a screenshot of their screen, that are close to the concept of taking a selfie in the voting booth.

Environmental impact

In a world facing resource limitations and a climate emergency that demands we minimize resource and energy usage, one might question the need for high-tech solutions online voting, and blockchain when paper-based systems are already functional. Also, there is a common misconception that blockchain technology inherently consumes astronomical amounts of energy. This popular belief was very likely widespread by Bitcoin's opponents. Bitcoin, indeed, wastes a huge quantity of electricity because of its Proof-of-Work consensus algorithm, which is based on the use of computational power (mining power, or hash rate) and incentivizes miners to spend electricity to secure the network. However, this is not the case for all use of blockchain or cryptocurrencies. Ethereum, for instance, has transitioned away from PoW to PoS consensus algorithm during an event called "The Merge" in September 2022. Since this event, the electricity consumption of the Ethereum network has decreased by an estimated 99.99% according to the Cambridge Blockchain Network Sustainability Index (CBNSI) [14] and currently consumes 2600 MWh and emits around 870 tons of CO2 annually [27].

Our system's environmental impact is expected to be minimal. As previously discussed in the cost estimation section, the majority of material requirements are already met, removing the need for additional hardware production. The network necessitates only a small number of peers for a limited duration, operated by organizations that already possess significant computational resources, and who would not need to acquire new machines. The few servers required for hosting the voting website or app, as well as the authentication service, are expected to handle a surge in demand over a short period. These services will likely be cloud-hosted, taking advantage of cloud providers' scalable solutions designed to manage activity peaks efficiently. Furthermore, the use of existing internet infrastructure and personal devices by voters means almost no extra hardware is needed. The total electricity consumption, amounting to a few computers operating for several hours and individuals using their devices for a few minutes, would be negligible in comparison to daily energy usage.

In comparison, the current system requires the acquisition and maintenance of thousands of e-voting machines, in addition to all materials required to host an election. Also, an important part of total emissions is due to the transportation of the millions of electors to the voting stations. We can therefore conclude that online elections would have a lower environmental impact than traditional ones. It remains to be seen, however, whether using the Internet for elections will not make us even more dependent on technology. But that is another debate.

Acceptation from the population

An important challenge to face regarding the implementation of online voting in Belgium is the acceptance from the population. Voices will likely be raised against such a system, for ideological, environmental, or democratic reasons. It will therefore be important to communicate that this system offers transparency while guaranteeing both the anonymity of votes and the accuracy of results. The complexity of cryptographic protocols may lead to a lack of trust among people. Also, the use of Zero-Knowledge Proofs might create an impression for some that the system operates like a black box, with unclear internal workings - which would be an incorrect assumption. For this reason, we believe that enabling users to verify their votes is a significant advantage, despite the absence of "receipt-freeness" not being the ideal outcome, according to the literature.

Democratic implications

If online voting proves to be more cost-effective than traditional paper or electronic voting methods, it could significantly impact our democratic processes. Reduced costs may lead to more frequent voting opportunities. In an era where calls for a 21st-century democratic system that leverages our interconnectedness are growing louder [69], online voting could facilitate referendums or public consultations on important issues.

Consider the 2023 Paris referendum on banning self-service electric scooters. Mayor Anne Hidalgo envisioned this as a democratic exercise, allowing citizens direct decision-making power. However, the outcome was a low turnout of just 7.46%, despite an overwhelming majority (89.03%) voting against the scooters [131]. The primary reason for this low participation was the requirement to vote in person, combined with minimal public interest in the topic. An online voting option might have encouraged more Parisians to participate.

Another instance is the prolonged government formation process post-elections, a recurring issue in Belgian politics. For example, in 2010-2011, it took 541 days to form a government. Some analysts predict similar challenges after the 2023 elections due to diverging party positions [60]. In such scenarios, quickly re-organized online elections could offer a feasible quicker solution to political stalemates.

The impact of online voting on turnout does not make consensus in the literature. After an analysis of seven elections in Estonia making use of online voting, it appears that Internet voting has not boosted turnout, which has remained stable over the years. An opinion shared by [135]. On the other hand, a study [18] showed that a simple parameter such as weather on the election day can have an impact on turnout, and also on the candidate choice. Online voting would reduce this last issue.

7.4 Future works

As outlined earlier, our system guarantees the ability for users to verify their votes, ensuring cast-as-intended verifiability. However, this capability leads to a compromise in receipt-freeness. Sandra Guasch and Paz Morillo, in their paper titled *How to challenge and cast your e-vote* [67], suggest a novel approach. Their method enables voters to both verify their actual vote and generate simulated proofs of alternative votes to mislead potential coercers. This approach could be explored in future work to achieve both cast-as-intended verifiability and coercion resistance. Alternatively, Kirillov *et al.*, who adapted the He and Su blind signature protocol in their work [86], propose utilizing ring signatures to ensure coercion-resistance in the voting system in future works.

The voting protocol used in our proposed model bases its cryptography on RSA to encrypt, decrypt, and sign data. However, it is important to note that RSA, like other popular public-key cryptosystems, is not quantum-resistant. RSA is based on the assumption that factoring large integers is computationally intractable. However, it could be broken by Shor's algorithm [121], an algorithm for quantum computers that can efficiently factorize large numbers. While current quantum computers, limited by their qubit capacity, are not yet advanced enough to break RSA, the rapid evolution in quantum computing technology presents a real risk of RSA becoming vulnerable in the coming years or decades. Given that our system stores all votes publicly and these records could be preserved until quantum computers can feasibly decrypt them, future works are needed to adapt the protocol with quantum-resistant cryptographic primaries.

Chapter 8

Conclusion

This thesis explored the topic of online elections. It proposed a model of a blockchain-based online voting system that was designed to be implemented in the context of Belgian elections.

First, a formal definition of the requirements of an election was realized in Section 4, and in particular the requirements specific to online voting. We synthesized what the literature said about them, and kept as essential requirements the integrity, anonymity, unreusability (one-human, one-vote), availability verifiability, and freeness for the voters. We thoroughly explored different voting systems, from the most traditional, using paper ballots, to the most modern, voting through the Internet. We analyzed how each requirement of an election was respected with paper voting, and highlighted its limitations. For electronic voting, we analyzed the two most frequent methods, either on direct-recording electronic voting machines (DRE) or on machines with voter-verified printed audit trails (VVPAT). We showed that the first was often criticized in the literature and that some of these machines have proven to be unsecured, as Halderman showed with the Diebold AccuVote-TS, used in the US in the mid-2000s. On the other hand, the second has gained more trust from the public, since it allows the voter to individually verify that their vote was encoded with the correct candidate, making it much closer to traditional paper-based voting, but with instantaneous count.

We have also analyzed different models of Internet voting systems, including tools for small-scale elections with low coercion levels, like Helios, and also national election frameworks, like the old and the new frameworks of Internet voting in Estonia, and the New South Wales' iVote system. For each model, the analysis was always directed to the respect of the requirements formalized in Section 4.1. We highlighted the risks brought by each system, in particular the risks of online voting which are largely discussed in the literature. We also discussed the notion of end-to-end verifiability and explained why it was a necessary property to ensure transparency.

After this comparison, we introduced how blockchain technology could solve some issues of online voting, such as protecting the integrity, securing the server by replacing it with a database on a distributed network and bringing more transparency.

Subsequently, in Section 5, we first introduced an example of a method of a blockchain-based voting system with a simple and naive smart contract that could handle an election, but with a big lack of privacy and anonymity. To solve this privacy problem, we proceeded to a literature review and compared 6 solutions of blockchain-based voting with distinct architecture. For each analyzed model, we first described the main characteristic, then analyzed whether each requirement was respected or not, and outlined the problems. We concluded this review with a comparison of the six models based on each requirement.

Finally, we proposed our own solution for a blockchain-based voting system in Belgium. We motivated our design choice and detailed the various aspects, including the network architecture, the organization involved, the authentication protocol, the cryptographic protocol to register voters, cast the votes, and tally the results, and described the process from the user's point of view. Our solution, detailed in Section

6, uses the Hyperledger Fabric Framework, an open-source permission blockchain network with state-of-the-art scalability performance. The network is composed of peers managed by partner organizations, selected meticulously based on their cybersecurity expertise, and also includes reader nodes for audibility and transparency. As it is a permissioned network, each node is authenticated via an authorized Certificate Authority. The list of authorized organizations and their respective CA are indicated in the channel policy. The ordering service - the nodes ordering the transactions in blocks into the ledger - uses a Byzantine-fault tolerant consensus protocol, that can continue to work as long as two-thirds of the nodes act honestly. Chaincodes manage the reception and the decryption of the votes. One chaincode (smart contract) is deployed for each constituency.

The cryptographic protocol used for the transmission and decryption of the votes is adapted from the works of He and Su [73] and Kirillov [86]. After successful authentication using itsme®, an eligible voter receives a certificate issued by $CA\_Electors$ and interacts with its designated chaincode, indicated in the certificate. The protocol uses blind signature and Idemix to guarantee anonymity.

We then analyzed this solution considering the requirements we formalized at the beginning and used all along the thesis. We have highlighted that it respected the requirements of integrity, authenticity, un-reusability, anonymity, freeness, auditability, verifiability, and availability, but it fell short in receipt-freeness. The eligibility remains the election authority's responsibility, to establish lists of eligible voters, and not to issue certificates to unexisting or unauthorized voters. It also respected the requirement specifics for Belgian elections, as it keeps track of the user who authenticated (as the vote is mandatory), handles blank and preference votes, and allows to vote on multiple elections on the same day in a straightforward process. The voters have the choice to vote either online or on-site. A database, synchronized with the polling station, ensures a voter cannot vote twice. As each chaincode handles one constituency (such as a municipality), the system could be first tested in pilot municipalities before being expanded further.

In the security analysis, we mentioned that the model could enhance security compared to an online voting system on a centralized server, but it cannot solve every problem related to online voting. There currently exists no perfect solution for i-voting. It is a domain of research still in development. We have shown an example where blockchain can bring improvements, but at the cost of concessions (such as publicly storing all ballots forever) and not fulfilling the requirements of receipt-freeness. Moreover, some risks inherent in online voting simply can't be solved by blockchain.

In the discussion in Section 7.3, we conducted a comparative analysis of the costs associated with implementing such a system in contrast to current methods. Our conclusion posited that online voting could become more cost-effective once the initial development costs have been amortized. Furthermore, we provided insights into the environmental impact, legal considerations, and democratic implications of the proposed solution. We argued that the solution would simplify the voting process for citizens who cannot get physically to the polling station (due to health problems, unavailability, or living or traveling abroad. It would also open the door to more participatory democracy, by offering the opportunity to organize votes more easily (such as elections, referendums, or public consultation).

Appendix A

A.1 Basic Ballot Smart Contract

```
1  pragma solidity >=0.7.0 <0.9.0;
2
3  /* @dev Implements voting process*/
4  contract Ballot {
5
6      struct Voter {
7          uint weight; // weight could be accumulated by delegation
8          bool voted;  // if true, that person already voted
9      }
10
11     struct Proposal {
12         bytes32 name;   // short name (up to 32 bytes)
13         uint voteCount; // number of accumulated votes
14     }
15
16     address public chairperson;
17
18     mapping(address => Voter) public voters;
19
20     Proposal[] public proposals;
21
22     /**
23      * @dev Create a new ballot to choose one of 'proposalNames'.
24      * @param proposalNames names of proposals
25      */
26     constructor(bytes32[] memory proposalNames) {
27         chairperson = msg.sender;
28         voters[chairperson].weight = 1;
29
30         for (uint i = 0; i < proposalNames.length; i++) {
31             // Creates a temporary Proposal object and
32             //appends it to the end of 'proposals'.
33             proposals.push(Proposal({
34                 name: proposalNames[i],
35                 voteCount: 0
36             }));
37         }
38     }
39
40     /**
41      * @dev Give 'voter' the right to vote on this ballot. May only
           be called by 'chairperson'.
42      * @param voter address of voter
43      */
44     function giveRightToVote(address voter) public {
45         require(
46             msg.sender == chairperson,
47             "Only chairperson can give right to vote."
48         );
49         require(
50             !voters[voter].voted,
51             "The voter already voted."
52         );
53         require(voters[voter].weight == 0);
```

```
54          voters[voter].weight = 1;
55      }
56
57      /**
58       * @dev Give your vote to proposal 'proposals[proposal].name'.
59       * @param proposal index of proposal in the proposals array
60       */
61      function vote(uint proposal) public {
62          Voter storage sender = voters[msg.sender];
63          require(sender.weight != 0, "Has no right to vote");
64          require(!sender.voted, "Already voted.");
65          sender.voted = true;
66
67          proposals[proposal].voteCount += sender.weight;
68      }
69
70      /**
71       * @dev Computes the winning proposal taking all previous votes
             into account.
72       * @return winningProposal_ index of winning proposal in the
             proposals array
73       */
74      function winningProposal() public view
75              returns (uint winningProposal_)
76      {
77          uint winningVoteCount = 0;
78          for (uint p = 0; p < proposals.length; p++) {
79              if (proposals[p].voteCount > winningVoteCount) {
80                  winningVoteCount = proposals[p].voteCount;
81                  winningProposal_ = p;
82              }
83          }
84      }
85
86      /**
87       * @dev Calls winningProposal() function to get the index of the
             winner contained in the proposals array and then
88       * @return winnerName_ the name of the winner
89       */
90      function winnerName() public view
91              returns (bytes32 winnerName_)
92      {
93          winnerName_ = proposals[winningProposal()].name;
94      }
95  }
```

Listing A.1: A basic ballot Smart Contract in Solidity. MIT License

Bibliography

[1] The potential consortium launched to pilot the new european digital identity wallet (eudiw), 10.7.2023.

[2] Electoral commissioner's determination - ivote will not be used for 2023 nsw state election. `https://elections.nsw.gov.au/about-us/media-centre/news-and-media-releases/electoral-commissioner-ivote-determination`, 13 July 2023.

[3] *The Current State and Issue on the Implementation of Electronic Voting System in Internet*, 2002.

[4] Press release no. 19/2009. use of voting computers in 2005 bundestag election unconstitutional. Mar 2009.

[5] 1er juin 2022. - loi modifiant la loi du 23 mars 1989 relative à l'élection du parlement européen en vue d'offrir aux citoyens la faculté de voter dès l'âge de 16 ans. *Moniteur belge*, 2022032290, 28/06/2022.

[6] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, page 335–348. USENIX Association, 2008.

[7] Ben Adida, Olivier Pereira, and Jean-Jacques Quisquater. Electing a university president using open-audit voting: analysis of real-world use of helios. *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, 07 2009.

[8] Agora. Whitepaper. Technical report, 2017.

[9] Hosam Alamleh and A. A. AlQahtani. Analysis of the design requirements for remote internet-based e-voting systems. *2021 IEEE World AI IoT Congress (AIIoT)*, pages 0386–0390, 2021.

[10] Syed Taha Ali and Judy Murray. An overview of end-to-end verifiable voting systems. 2016.

[11] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, 2019.

[12] Arne Ansper, Ahto Buldas, Aivo Jürgenson, Mart Oruaas, Jaan Priisalu, Kaido Raiend, Anto Veldre, Jan Willemson, and Kaur Virunurm. E-voting concept security: analysis and measures. Technical report, Estonia's National Electoral Committee, 2010.

[13] Trueb Baltic AS. Estonian electronic id–card application specification. Technical report, AS Sertifitseerimiskeskus, 2014.

[14] Cambridge Digital Assets Programme (CDAP) Team at the Cambridge Centre for Alternative Finance. Cambridge blockchain network sustainability index (cbnsi). `https://ccaf.io/cbnsi/ethereum/ethereum_merge`.

[15] Adam Back. Hashcash - a denial of service counter-measure. 09 2002.

[16] Abdul Based. Security aspects of internet based voting. In Tarek Sobh, Khaled Elleithy, and Ausif Mahmood, editors, *Novel Algorithms and Techniques in Telecommunications and Networking*, pages 329–332, Dordrecht, 2010. Springer Netherlands.

[17] Md. Abdul Based and Stig Fr. Mjølsnes. Security requirements for internet voting systems. In Tarek Sobh and Khaled Elleithy, editors, *Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering*, pages 519–530, New York, NY, 2013. Springer New York.

[18] Anna Bassi. Weather, risk, and voting: An experimental analysis of the effect of weather on vote choice. *Journal of Experimental Political Science*, 6(1):17–32, 2019.

[19] Ali Benabdallah, Antoine Audras, Louis Coudert, Nour El Madhoun, and Mohamad Badra. Analysis of blockchain solutions for e-voting: A systematic literature review. *IEEE Access*, 10:70746–70759, 2022.

[20] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, page 544–553, New York, NY, USA, 1994. Association for Computing Machinery.

[21] Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, USA, 1987. AAI8809191.

[22] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 103–112, New York, NY, USA, 1988. Association for Computing Machinery.

[23] Katharina Bräunlich and Rüdiger Grimm. Development of a formal security model for electronic voting systems. *Int. J. Inf. Secur. Priv.*, 7:1–28, 2013.

[24] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. 2014.

[25] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[26] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association.

[27] Crypto Carbon Ratings Institute (CCRI). The merge – implications on the electricity consumption and carbon footprint of the ethereum network. Sep. 2022.

[28] Brad Chase and Ethan MacBrough. Analysis of the xrp ledger consensus protocol, 2018.

[29] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US.

[30] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, feb 1981.

[31] Guang Chen, Bing Xu, Mingdong Lu, and Nian-Shing Chen. Exploring blockchain technology and its potential applications for education. *Smart Learning Environments*, 5:1–10, 2018.

[32] Dylan Clarke and Tarvi Martens. E-voting in estonia. *CoRR*, abs/1606.08654, 2016.

[33] Danny De Cock. "introducing belgian eid cards", 2013.

[34] Cour constitutionnelle. Arrêt n° 116/2023 du 20 juillet 2023. 20 juillet 2023.

[35] Véronique Cortier, Pierrick Gaudry, Stéphane Glondu, and Sylvain Ruhault. French 2022 legislatives elections: a verifiability experiment. In *The E-Vote-ID Conference 2023*, Luxembourg City, Luxembourg, October 2023.

[36] Wim Coulier. itsme practice statement version 2.4, 2023.

[37] Wei Dai. B-money. `http://www.weidai.com/bmoney.txt`, 1998. Accessed: 2023-12-05.

[38] Régis Dandoy. The impact of e-voting on turnout: Insights from the belgian case. pages 29–37, 04 2014.

[39] Olawande Daramola and Darren Thebus. Architecture-centric evaluation of blockchain-based smart contract e-voting for national elections. *Informatics*, 7(2):16, May 2020.

[40] Marianne Dengo. Blockchain voting: A systematic literature review. Master's thesis, University of Tartu, 2020.

[41] SPF Intérieur Direction des Elections. Electeurs. `https://elections.fgov.be/electeurs`.

[42] Gawade Dinesh R., Shirolkar Amardeep A., and Patil Sagar R. E-voting system using mobile sms. *IJRET: International Journal of Research in Engineering and Technology*, 2015.

[43] Evan Duffield and Daniel Diaz. Dash: A privacy-centric crypto-currency. `https://github.com/dashpay/dash/wiki/Whitepaper`, 2014. Accessed: 2023-12-05.

[44] Susan Dzieduszycka-Suinat, Ir. M. Menco B. Ph., Joseph Kiniry, D. M. Zimmerman, Daniel Wagner, Philip Robinson, and Adam. The future of voting end-to-end verifiable internet voting specification and feasibility assessment study. 2015.

[45] Collège d'experts. Rapport du collège d'experts chargés du contrôle des systèmes électroniques de vote, de dépouillement et de collecte des résultats Élections simultanées du 26 mai 2019 pour le parlement européen, la chambre des représentants et les parlements de région et communauté. Technical report, Service Public Fédéral Intérieur, Direction des Elections, 7 Juin 2019.

[46] Sam Edwards. Spanish police seize ballot boxes in catalan referendum. `https://www.reuters.com/article/uk-spain-politics-catalonia-clash-idUKKCN1C611L/`, Oct. 1st 2017.

[47] Piret Ehin, Mihkel Solvak, Jan Willemson, and Priit Vinkel. Internet voting in estonia 2005–2019: Evidence from eleven elections. *Government Information Quarterly*, 39(4):101718, 2022.

[48] Regulation (eu) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec. *OJ*, L 257:73–114, 28.8.2014.

[49] Willems Emmanuel, Markowitch Olivier, Van Geyt Karel, Dossogne Jérôme, Dumortier Fabrice, and Jean-Michel Dricot. Rapport du collège d'experts chargés du contrôle des systèmes de vote électronique pour les élections communales de la région de bruxelles-capitale. Technical report, Parlement de la Région de Bruxelles-Capitale, 14 octobre 2018.

[50] Chantal Enguehard. Analyse des vulnérabilités de trois modes de vote à distance. *Legalis.net*, 3:13–31, September 2008.

[51] Aleksander Essex. Internet voting in canada: A cyber security perspective. *Department of Electrical and Computer Engineering*.

[52] Eurostat. Internet access of individuals, 2010 and 2023. `https://ec.europa.eu/eurostat/databrowser/view/isoc_ci_ifp_iu/default/table`.

[53] Hyperledger Farbic. Official documentation. `https://hyperledger-fabric.readthedocs.io/en/latest/index.html`. Accessed: 2023-12-23.

[54] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. Security analysis of the diebold accuvote-ts voting machine. In *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, EVT'07, page 2, USA, 2007. USENIX Association.

[55] Tamara Finogina and Javier Herranz. On remote electronic voting with both coercion resistance and cast-as-intended verifiability. *Journal of Information Security and Applications*, 76:103554, 2023.

[56] NIST FIPS. Secure hash standard (shs), federal inf. process. stds.

[57] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, pages 244–251, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[58] S.S. Gandhi, A.W. Kiwelekar, L.D. Netak, and H.S. Wankhede. Security requirement analysis of blockchain-based e-voting systems. *Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, 131, 2023.

[59] Thibault Gaudin, Vincent Jacquet, Jean-Benoit Pilet, and Min Reuchamps. Consultation populaire et référendum en belgique. *Courrier hebdomadaire*, 25(2390-2391):5–62, 2018.

[60] Jules Gheude. Pourquoi la belgique risque fort d'être ingouvernable après les élections de 2024. `https://www.lalibre.be/debats/opinions/2023/09/22/pourquoi-la-belgique-risque-fort-detre-ingouvernable-apres-les-elections-de-2024` 22th Sep. 2023.

[61] John Paul Gibson, Vanessa Teague, Robert Krimmer, and Julia Pomares. A review of E-voting: the past, present and future. *Annals of Telecommunications - annales des télécommunications*, 71(7):279 – 286, August 2016.

[62] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, jun 2002.

[63] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof system. *SIAM Journal on Computing*, 18:186–208, 1985.

[64] Sri Nikhil Gupta Gourisetti, Michael Mylrea, and Hari Patangia. Evaluation and demonstration of blockchain applicability framework. *IEEE Transactions on Engineering Management*, 67:1142–1156, 2020.

[65] Dimitris A Gritzalis. Principles and requirements for a secure e-voting system. *Computers & Security*, 21(6):539–556, 2002.

[66] Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 467–482, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[67] Sandra Guasch and Paz Morillo. How to challenge and cast your e-vote. pages 130–145, 05 2017.

[68] Ye Guo and Chen Liang. Blockchain application and outlook in the banking industry. *Financial Innovation*, 2:1–12, 2016.

[69] Antonin Guyader. Vers une démocratie plus numérique. *Pouvoirs*, 2018.

[70] Stuart Haber, Josh Benaloh, and Shai Halevi. The helios e-voting demo for the iacr. Technical report, International Association for Cryptographic Research, 2010.

[71] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.

[72] J. Alex Halderman. Practical attacks on real-world e-voting. *Real-World Electronic Voting Design, Analysis and Deployment, Chapter 7*, 2017.

[73] O He and Zhongmin Su. *A new practical secure e-voting scheme.* na, 1998.

[74] Sven Heiberg, Kristjan Krips, and Jan Willemson. *Mobile Voting – Still Too Risky?*, pages 263–278. 09 2021.

[75] Jens M. Helbig. *De la blockchain à crypto-investisseur.* KHLE finance, 2019.

[76] Friorik Hjalmarsson, Gunnlaugur Hreioarsson, Mohammad Hamdaqa, and Gisli Hjálmtýsson. Blockchain-based e-voting system. pages 983–986, 07 2018.

[77] Itsme. Itsme openid connect api documentation. `https://belgianmobileid.github.io/doc/authentication/`.

[78] Uzma Jafar, Mohd Juzaiddin Ab Aziz, and Zarina Shukur. Blockchain for electronic voting system—review and open research challenges. *Sensors*, 21(17), 2021.

[79] Rui Joaquim, Paulo Ferreira, and Carlos Ribeiro. Eviv: An end-to-end verifiable internet voting system. *Computers & Security*, 32:170–191, 2013.

[80] Quentin Kaiser. Shedding some light on the new belgian evoting system. `https://quentinkaiser.be/evoting/2018/06/18/shedding-light-belgian-evoting/`, Jun 18, 2018.

[81] K. Kaliyamurthie, R. Udayakumar, D. Parameswari, and S. Mugunthan. Highly secured online voting system over network. *Indian journal of science and technology*, 6:4831–4836, 2013.

[82] Anamika Kannoly, Samiksha Arun, Ganesan Subramanian, S.K.Pandey, and M.Arumugaselvi. Impact of blockchain in the voting system. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 9, July 2020.

[83] Hyunyeon Kim, Kyung Eun Kim, Soohan Park, and Jongsoo Sohn. E-voting system using homomorphic encryption and blockchain technology to encrypt voter data. *CoRR*, abs/2111.05096, 2021.

[84] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. `https://peercoin.net/assets/paper/peercoin-paper.pdf`, 2012. Accessed: 2023-12-05.

[85] Denis Kirillov, Vladimir Korkhov, Vadim Petrunin, and Mikhail Makarov. Performance of the secret electronic voting scheme using hyperledger fabric permissioned blockchain. In *Computational Science and Its Applications – ICCSA 2020*, pages 25–36, Cham, 2020. Springer International Publishing.

[86] Vladimir Kirillov, Denis nd Korkhov, Vadim Petrunin, Mikhail Makarov, Ildar M. Khamitov, and Victor Dostov. Implementation of an e-voting scheme using hyperledger fabric permissioned blockchain. In *Computational Science and Its Applications – ICCSA 2019*, pages 509–521, Cham, 2019. Springer International Publishing.

[87] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security – ESORICS 2010*, pages 389–404, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[88] Robert Krimmer, David Duenas-Cid, and Iuliia Krivonosova. New methodology for calculating cost-efficiency of different ways of voting: is internet voting cheaper? *Public Money & Management*, 41(1):17–26, 2021.

[89] Mahender Kumar, Satish Chand, and C. P. Katti. A secure end-to-end verifiable internet-voting system using identity-based blind signature. *IEEE Systems Journal*, 14(2):2032–2041, 2020.

[90] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982.

[91] Wouter Lueks, Iñigo Querejeta-Azurmendi, and Carmela Troncoso. VoteAgain: A scalable coercion-resistant voting system. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1553–1570. USENIX Association, August 2020.

[92] Ülle Madise and Priit Vinkel. *Internet Voting in Estonia: From Constitutional Debate to Evaluation of Experience over Six Elections*, pages 53–72. Springer International Publishing, Cham, 2014.

[93] Gupta Mayank, Kumar Manish, Sharma Devendra Pratap, and Kumar Raj. The risks of remote voting. *Ijraset Journal For Research in Applied Science and Engineering Technology*, 11, 2023.

[94] Patrick Mccorry, Maryam Mehrnezhad, Ehsan Toreini, Siamak F. Shahandashti, and Feng Hao. On secure e-voting over blockchain. *Digital Threats*, 2(4), oct 2021.

[95] Kashif Mehboob, Junaid Arshad, and Muhammad Khan. Secure digital voting system based on blockchain technology. *International Journal of Electronic Government Research*, 14:53–62, 01 2018.

[96] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979.

[97] Ahmed Afif Monrat, Olov Schelén, and Karl Andersson. A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151, 2019.

[98] Teógenes Moura and A. Gomes. Blockchain voting and its effects on election transparency and voter confidence. In *Proceedings of the 18th Annual International Conference on Digital Government Research*, 2017.

[99] Malik Hamza Murtaza, Zahoor Ahmed Alizai, and Zubair Iqbal. Blockchain based anonymous voting system using zksnarks. In *2019 International Conference on Applied and Engineering Mathematics (ICAEM)*, pages 209–214, 2019.

[100] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008.

[101] Antonio Nappa, Christopher Hobbs, and Andrea Lanzi. Deja-vu: A glimpse on radioactive soft-error consequences on classical and quantum computations. *CoRR*, abs/2105.05103, 2021.

[102] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, page 116–125, New York, NY, USA, 2001. Association for Computing Machinery.

[103] Vehbi Neziri, Isak Shabani, Ramadan Dervishi, and Blerim Rexha. Assuring anonymity and privacy in electronic voting with distributed technologies based on blockchain. *Applied Sciences*, 12(11):5477, May 2022.

[104] Maina Olembo, Patrick Schmidt, and Melanie Volkamer. Introducing verifiability in the polyas remote electronic voting system. pages 127 – 134, 09 2011.

[105] OSCE/ODIHR. Odihr election expert team final report - estonia, parliamentary elections, 3 march 2019. Technical report, Office for Democratic Institutions and Human Rights.

[106] Sunoo Park, Michael Specter, Neha Narula, and Ronald L. Rivest (MIT). Going from bad to worse: From internet voting to blockchain voting. *Journal of Cybersecurity*, 7, 2021.

[107] Ahmed Yusuf Patel, Cornelius Johannes Kruger, and Henri Emil van Rensburg. Why proprietary blockchains are not suitable for online voting! *Published Online by the SAICSIT 2023 Organising Committee Potchefstroom: South African Institute of Computer Scientists & In-formation Technologists*, page 30, 2023.

[108] Guido Perboli, Stefano Musso, and Mariangela Rosano. Blockchain in logistics and supply chain: A lean approach for designing real-world use cases. *IEEE Access*, 6:62018–62028, 2018.

[109] Christophe Petit and Liran Lerman. Protocols, cryptanalysis and mathematical cryptology (info-f514) "introduction - e-voting".

[110] Maksym Petkus. Why and how zk-snark works, 2019.

[111] Jared Polites. The first blockchain-backed presidential election just took place in sierra leone. *CNN*, 2021.

[112] Rahul Puniani. Conceptualization of a blockchain based voting ecosystem in estonia. Master's thesis, University of Tartu, 2020.

[113] Ghassan Z. Qadah and Taha Rani. Electronic voting systems: Requirements, design, and implementation. *Computer Standards & Interfaces*, 2007.

[114] Igor Radanovic and Robert Likić. Opportunities for use of blockchain technology in medicine. *Applied Health Economics and Health Policy*, 16:583–590, 2018.

[115] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[116] Peter Y. A. Ryan, David Bismark, James Heather, Steve Schneider, and Zhe Xia. PrÊt À voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.

[117] Rihab H Sahib and Eman S. Al-Shamery. A review on distributed blockchain technology for e-voting systems. *Journal of Physics: Conference Series*, 1804(1):012050, feb 2021.

[118] M. Santhosh, 2. S. Kavitha, .. Keerthana, L. Suganya, and .. S. Krishnakumar. Electronic voting machine using internet. *International Journal of communication and computer Technologies*, 2019.

[119] Direction des Elections Service public fédéral Intérieur. Les votes blancs vont-ils à la majorité? `https://elections.fgov.be/node/111327`. Accessed: October 1st 2023.

[120] Basit Shahzad and Jon Crowcroft. Trustworthy electronic voting using adjusted blockchain technology. *IEEE Access*, 7:24477–24488, 2019.

[121] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1995.

[122] Vivek Singh Sisodiya and Hitendra Garg. A comprehensive study of blockchain and its various applications. In *2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC)*, pages 475–480, 2020.

[123] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. Security analysis of the Estonian Internet voting system. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*. ACM, November 2014.

[124] Christian Stoll, Lena Klaaßen, and Ulrich Gallersdörfer. The carbon footprint of bitcoin. *Joule*, 3(7):1647–1661, 2019.

[125] Florence Susant. Le vote électronique - une tempête dans un verre d'eau. *Au Quotidien*, 2014.

[126] Nick Szabo. Bit gold. `http://unenumerated.blogspot.com/2005/12/bit-gold.html`, 2005. Accessed: 2023-12-05.

[127] Evrim Tan, Ellen Lerouge, Jan Du Caju, and Daniël Du Seuil. Verification of education credentials on european blockchain services infrastructure (ebsi): Action research in a cross-border use case between belgium and italy. *Big Data and Cognitive Computing*, 7(2), 2023.

[128] Antti Tani. Zero-knowledge proofs in blockchain applications. Master's thesis, University of Helsinki, 2020.

[129] Ruhi Taş and Ömer Özgür Tanrıöver. A systematic review of challenges and opportunities of blockchain for e-voting. *Symmetry*, 12(8), 2020.

[130] Agora Blockchain Team. Agora official statement regarding sierra leone election. *Medium*, Mar 2018.

[131] Laurent Telo. Vote sur les trottinettes électriques en libre-service à paris : Anne hidalgo se réjouit de la victoire écrasante du « contre », l'opposition de droite dénonce « un flop ». `https://www.lemonde.fr/politique/article/2023/04/03/trottinettes-electriques-en-libre-service-a-paris-anne-hidalgo-se-rejouit-de-la-6168012_823448.html`, 3rd Apr. 2023.

[132] Jean-Benoit Pilet (ULB), Bart Preneel (KUL), Silvia Erzeel (VUB), Olivier Pereira (UCL), Fanny Sbaraglia (ULB), Aurélie Tibbaut (ULB), Xavier Carpent (KUL), and Régis Dandoy (ULB). Etude sur la possibilité

d'introduire le vote internet en belgique, volets 1 et 2. Technical report, Université libre de Bruxelles, Katholieke Universiteit Leuven, Vrije Universiteit Brussel, Université Catholique de Louvain, Policy Lab SciencePo ULB, 2021.

[133] Valimised.ee. *Voter turnout at Riigikogu elections was 63.7%*. Mar 2023.

[134] Nicolas van Saberhagen. Cryptonote v 2.0. `https://cryptonote.org/whitepaper.pdf`, 2013. Accessed: 2023-12-05.

[135] Kristjan Vassil and Till Weber. A bottleneck model of e-voting: Why technology fails to boost turnout. *New Media & Society*, 13(8):1336–1354, 2011.

[136] Carlos Vegas. The new belgian e-voting system. In *Electronic Voting*, 2012.

[137] Adolfo Villafiorita, Komminist Weldemariam, and Roberto Tiella. Development, formal verification, and evaluation of an e-voting system with vvpat. *IEEE Transactions on Information Forensics and Security*, 4(4):651–661, 2009.

[138] Maria-Victoria Vladucu, Ziqian Dong, Jorge Medina, and Roberto Rojas-Cessa. E-voting meets blockchain: A survey. *IEEE Access*, 11:23293–23308, 2023.

[139] Hyperledger Whitepaper. An introduction to hyperledger. `https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/Hyperledger/Offers/HL_Whitepaper_IntroductiontoHyperledger.pdf`, 2018.

[140] Scott Wolchok, Eric Wustrow, Dawn Isabel, and J. Alex Halderman. Attacking the washington, d.c. internet voting system. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security*, pages 114–128, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[141] Peng Zhuang, Talha Zamir, and Hao Liang. Blockchain for cybersecurity in smart grid: A comprehensive survey. *IEEE Transactions on Industrial Informatics*, 17:3–19, 2021.

[142] Ilya Zikratov, Alexey Kuzmin, Vladislav Akimenko, Viktor Niculichev, and Lucas Yalansky. Ensuring data integrity using blockchain technology. In *2017 20th Conference of Open Innovations Association (FRUCT)*, pages 534–539, 2017.